НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

В.о.завідувача кафедри

_____ Оксана ТИМОЩУК

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Система обробки і класифікації елктрокардіограм»

Виконав:
студент IV курсу, групи КА-64
Лозовий Кирило Сергійович _____

Керівник:
професор кафедри ММСА, д.т.н. Данилов В. Я. _____

Консультант з економічного розділу:
доцент, к.е.н. Шевчук О. А. _____

Консультант з нормконтролю:
доцент, к.е.н. Коваленко А.Є. _____

Рецензент:
доцент кафедри системного проектування
ІПСА, к.т.н. Г.Д. Кисельов _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

В.о.завідувача кафедри

_____ Оксана ТИМОЩУК

«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Лозовий Кирило Сергійович

1. Тема роботи «**Система обробки і класифікації елктрокардіограм**», керівник роботи професор кафедри ММСА, д.т.н. Данилов В. Я., затверджені наказом по університету від «_25_»_травня_ 20_20_ р. №_1143-с__

2. Термін подання студентом роботи 08 травня 2020 року _____

3. Вихідні дані до роботи

1. Операційна система Windows 10
2. Частота процесора 2.9 ГГц
3. Мова програмування Python 3
4. Середовище розробки – JetBrains PyCharm
5. Бібліотеки, що використовувалися: Pandas, Sklearn, Keras, Tensorflow, Matplotlib, NumPy

4. Зміст роботи

1. Проаналізувати існуючі математичні моделі

2. Проаналізувати моделі побудови нейронних мереж CNN
3. Розробити систему оцінки побудованої моделі
4. Виконати економічний аналіз програмного продукту

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Презентація

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	к.е.н., доц. Шевчук О. А.	21.04.20	28.05.20

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	13.04.20	
2	Аналіз технологій	21.04.20	
3	Підбір найкращої моделі	29.04.20	
4	Обробка вхідної вибірки	5.05.20	
5	Навчання моделі, коригування вхідних даних і параметрів	9.05.20	
6	Тестування продукту, отримання результатів	11.05.20	
7	Оформлення дипломної роботи	18.05.20	

Студент Лозовий К.С.

Керівник Данилов В.Я.

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломної роботи.

РЕФЕРАТ

Дипломна робота: 55 с., 23 рис., 13 табл., 4 додатки, 25 джерел.

ЕЛЕКТРОКАРДІОГРАМА, НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, ОБРОБКА СИГНАЛІВ.

Об’єкт дослідження – датасет з кардіограмами здорових і хворих людей.

Мета роботи – створення системи обробки і класифікації кардіограм для виявлення захворювань у людей по ЕКГ.

Предмет дослідження –методи обробки вхідного сигналу електрокардіограми і класифікація обробленого сигналу нейронною мережею.

Представлена система може використовуватися для визначення наявності хвороб серця у людини.

ABSTRACT

The theme: “Cardiogram processing and classification system”

Diploma: 55 pages, 23 figures, 13 tables, 4 appendices, 25 sources.

ELECTROCARDIOGRAPHY, NEURAL NETWORKS, MACHINE LEARNING, SIGNAL PROCESSING.

The object of the study is a dataset with cardiograms of healthy and sick people.

The purpose of work - creation of system of processing and classification of cardiograms for detection of diseases at people on an ECG.

The subject of research - methods of processing the input signal of the electrocardiogram and the classification of the processed signal by the neural network.

The presented system can be used to determine the presence of heart diseases.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

CNN – згорткова нейронна мережа

RNN – рекурентна нейронна мережа

ЕКГ – електрокардіограма

ФР – фрактальна розмірність

ШІ – штучний інтелект

ШНМ – штучні нейронні мережі

ВСТУП

Серцево-судинні захворювання - це клас захворювань, які можуть розвиватися неявно протягом усього життя і прогресувати в хронічну стадію на той час, коли з'являються перші симптоми. ССЗ займають перше місце серед усіх причин смертності населення: на них припадає 56,7% всіх смертей, щорічно в Україні від них помирає більше 400 тис. людей.

Відповідно до міжнародних тенденцій і стратегією розвитку медичної науки в Україні, на сьогоднішній день значна увага приділяється розвитку профілактичної медицини, розробці та впровадженню нових ефективних методів і засобів запобігання захворюванням, зміцненню здоров'я дітей та працюючого населення.

Більшість клінічних досліджень захворювань серцево-судинної системи засновані на аналізі електрокардіограм (ЕКГ) і вивченні ряду інших реєстрованих сигналів, що ілюструють біоелектричну активність серця. До безперечних переваг такого підходу можна віднести відносну простоту, доступність, неінвазивність і достатньо високу інформативність. ЕКГ - функціональний метод дослідження, суть якого полягає у визначенні стану серця серцево-судинної системи щодо змін в його електричної активності. Цей метод дослідження на сьогоднішній день є найпоширенішим і проводиться практично у всіх медичних закладах.

Основна проблема електрографічного методу діагностики захворювань полягає в тому, що традиційні методи аналізу електрокардіограм не завжди дозволяють діагностувати серцеві захворювання з високою вірогідністю. Часто достатньо серйозні серцеві захворювання відображаються на ЕКГ лише незначною зміною амплітуди і форми піків. У багатьох випадках точність діагнозу залежить від досвіду і рівня класифікації лікаря. Щоб виключити людський фактор та автоматизувати аналіз ЕКГ, і знайти такий метод, який був би здатний розпізнавати найбільш характерні зміни ЕКГ при тих чи інших захворюваннях, з урахуванням того, що навіть при одному і тому ж

захворюванні ЕКГ можуть відрізнятися один від одного. Варіантом вирішення даної проблеми є використання нейронних мереж.

Нейронна мережа - набір математичних і алгоритмічних методів для вирішення широкого кола завдань. Мережа обробляє вхідну інформацію і подає на вихід свій прогноз для вирішення поставленої задачі.

Метою цієї роботи є створення нейронної мережі здатної розпізнавати, а також класифікувати серцево-судинні захворювання.

РОЗДІЛ 1 АНАЛІЗ ОБ’ЄКТУ ДОСЛІДЖЕННЯ

1.1 Аналіз вимог до системи розпізнавання ЕКГ. Постановка задачі

Метою дипломної роботи є створення системи обробки і класифікації кардіограм. Модель має спочатку обробити вхідні дані ЕКГ, потім виявити наявність серцевого захворювання у людини і класифікувати його.

Запропонована система може бути використана у медичній сфері, для швидкого і чіткого аналізу кардіограм у медичних закладах, а також для мобільних ЕКГ-пристроях.

Для досягнення мети дипломної роботи будуть виконані такі кроки:

- Розглянути предметну область(електрокардіограма, будова і робота серця)
- Розглянути сучасні методи обробки ЕКГ
- Розглянути методи навчання нейронних мереж і вибрати найкращий для поставленої мети
- Знайти і обробити дані для навчання нейронної мережі
- Провести навчання і тренування моделі
- Протестувати фінальну версію програми

1.2 Електрокардіографія. Будова і робота серця. Зв'язок між ними.

Електрокардіографія — графічна інтерпретація роботи серцевого м’язу. Криву, яка відображає електричну активність серця, називають електрокардіограмою. Електричний імпульс формується в синусовому вузлі через рівні проміжки часу. Середня швидкість роботи серця у здорової людини приймає значення від 60 до 90 ударів за хвилину. Оскільки серце розташоване всередині тіла у рідкій середовищі, що містить електроліти, шлях серцевого імпульсу може бути записаний, застосувавши електроди у певних положеннях поверхні тіла. Утворення імпульсу та його передача в міокарді виробляють слабкі електричні струми, які збільшуються на ЕКГ-машині, і запис може здійснюватися в рухомому графічному папері. Папір для запису має графічний напис великих та малих квадратів і рухається зі швидкістю 25 мм / секунд або

50 мм / секунд для запису ЕКГ. Кожен маленький квадрат становить 0,04 секунд (1/25 секунд), а кожен великий квадрат - 0,2 (1/5 секунд). Отже, ЕКГ являється графічною інтерпретацією різниці потенціалів, виникаючих під час роботи серця.

ЕКГ являється базовим методом, який використовують при дослідженнях хвороби серця, таких як аритмія і гіпертрофія.

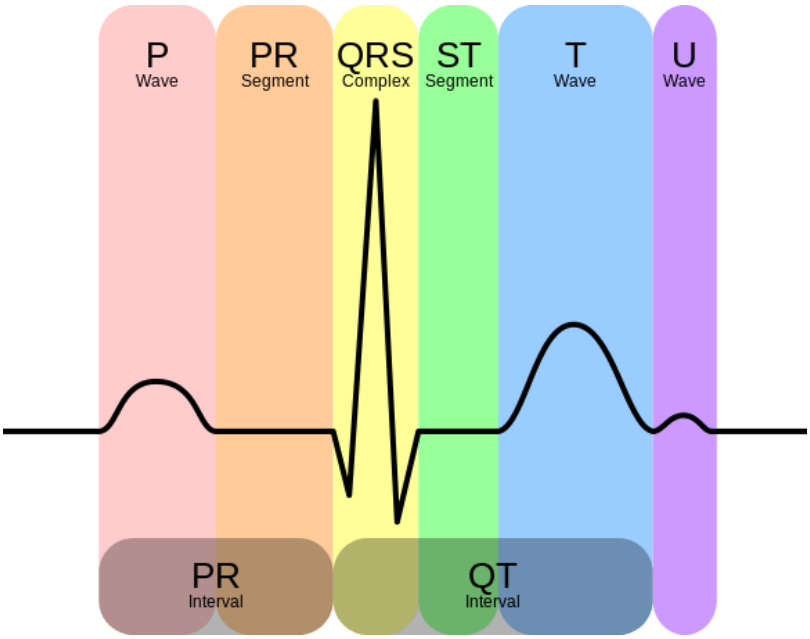


Рисунок 1.1 – Зубці та інтервали ЕКГ

Електрокардіограма являється дуже точним методом дослідження відхилень у роботі серця і може давати інформацію про локалізацію, глибину і час прояву, а також розповсюдженість хвороби.

На ЕКГ можна побачити інтервали та зубці (рис1.1), з яких і складається один серцевий цикл.

1) Зубець P відповідає деполяризації передсердь (максимум 0,12 секунд)

2) Інтервал P-Q — розповсюдження деполяризації до атриовентрикулярного вузла

3) Комплекс QRS об’єднує в собі три зубця – Q, R і S відповідно. Відображає збудження шлуночків серця. Q — перше негативне відхилення від ізоелектричної лінії. R — перше позитивне відхилення. S — негативне відхилення після R зубця

4) Сегмент S-T позначає початок фази деполяризації шлуночків

5) Зубець Т — хвиля деполяризації шлуночків

6) U хвиля. На кардіограмі ця хвиля часто не існує. З’являється лише при деяких порушеннях у роботі серця.

ЕКГ дає змогу діагностувати аритмію у хворого. Аритмія серця - патологічний стан, що приводить до порушення частоти, ритмічності і послідовності збудження і скорочення серця, тобто будь-який ритм серця, що відрізняється від нормального синусового ритму. При такому патологічному стані може істотно порушуватися нормальна скорочувальна активність серця, що, в свою чергу, може привести до цілого ряду серйозних ускладнень.(рис 1.2).

Аритмія найчастіше спочатку виявляється простими, але неспецифічними засобами: аускультацією серцебиття за допомогою стетоскопа або дослідженням периферичних імпульсів. Зазвичай вони не можуть діагностувати специфічну аритмію, але можуть загально вказувати на частоту серцевих скорочень та на регулярність чи нерегулярність. Не всі електричні імпульси серця виробляють відчутні удари - у багатьох серцевих аритміях передчасні або ненормальні удари не дають ефективної дії накачування і сприймаються як "пропущені" удари.

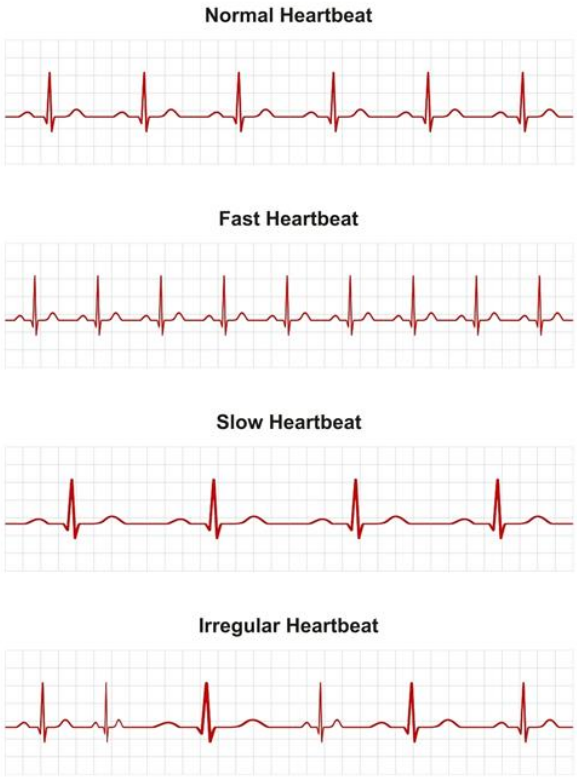


Рисунок 1.2 Нормальний серцевий ритм у порівнянні з аритмією різних типів

Монітор Холтера - це ЕКГ, записаний протягом 24 годин, щоб виявити аритмії, які можуть відбуватися за короткий період або непередбачувано протягом дня.

1.3 Аналіз методів обробки ЕКГ

Автоматичний аналіз ЕКГ- складний процес, який включає в себе чотири основних стадії:

- розпізнавання комплексів QRS на ЕКГ і визначення інтервалів RR;
- аналіз форми комплексів QRS, сегмента ST і зубця T;
- розпізнавання зубців P перед комплексами QRS і визначення інтервалів PQ;
- вироблення висновків про характер ритму і патологіях провідної системи серця на підставі отриманих даних.

Нижче наведемо інформацію про основні програми автоматичного аналізу ЕКГ.

- "CARDIMAX"

Це програмне забезпечення є опціональним, носієм є SD - карта. Воно може бути встановлено на холтеровські електрокардіографи компанії Fukuda Denshi, Японія, і володіє такими ключовими можливостями: одночасна реєстрація дванадцяти відведень, виявлення комплексів QRS, аналіз частоти скорочень, інтервалів PQ і QT, дослідження стану електричної осі, - інтерпретація і аналіз варіабельності серцевого ритму з формуванням висновку.

-“CardioDay”

Розробник - американська фірма GE Healthcare. “CardioDay” – це програмний продукт, створений для обробки і аналізу ЕКГ сигналу людини. Ця програма аналізує зміни у серцевому ритмі і фібриляції передсердь, вимірює інтервал QT між комплексами, аналізує функції кардіостимулятора, досліджує турбулентності серцевого ритму, а також автоматично інтерпретує отримані результати.

-“MEDILOG AR12 PLUS”

Програма вимірювань MEDILOG AR12 PLUS від компанії Schiller, Швейцарія, призначена для одноканальної або трьохканальної реєстрації електрокардіограм і пропонує широкий набір опцій аналізу ЕКГ: аналіз варіабельності серцевого ритму, аналіз сегментів QT і PQ, розпізнавання комплексів QRS, розпізнавання зубця PP і передсердний аналіз.

До недоліків даних програм можна віднести погану точність – близько 25% автоматичних інтерпретацій не збігаються з реальними даними. Також погана обробка шумів , і, як результат, неточні показники і прогнози.

1.4 Висновки до розділу

В цьому розділі було розглянуто існуючі програми для моніторингу і аналізу даних ЕКГ. Також було розглянуто основні складові ЕКГ сигналу та зв’язок між ними та роботою серця. На основі проаналізованих даних були розроблені вимоги до системи обробки і класифікації кардіограм.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Нейронні мережі

Штучні нейронні мережі, або конективістські — це обчислювальні системи, натхнені біологічними нейронними мережами, що складають мозок тварин. Описуючи такі системи часто вживають термін навчання(або тренування) нейронної мережі, який означає покращення точності прогнозу на кожній ітерації алгоритму. ШНМ використовують для обробки і розпізнавання зображень, сигналів, текстів, аудіо. Нейронній мережі не потрібно знати що саме вона шукає на зображенні чи у тексті, натомість вона створює свій власний набір характеристик і ознак, притаманних об’єкту дослідження. Штучна нейронна мережа складається з нейронів, які з’єднані між собою зв’язками, кожен з яких має власне значення. Кожен такий зв’язок передає значення від одного нейрону до іншого, і може змінюватися у процесі навчання.

Зазвичай нейрон у ШНМ приймає значення від 0 до 1, а значення на з’єднані між нейронами дійсним числом. Значення на виході нейрона обчислюється як лінійна комбінація значень нейронів і ваги на попередньому шарі. Штучні нейрони та з'єднання зазвичай мають вагу, яка підлаштовується в перебігу навчання. Вага збільшує або зменшує силу сигналу на з'єднанні. Штучні нейрони зазвичай організовано в шари. Різні шари можуть виконувати різні види перетворень своїх входів. Спочатку дані подаються на вхідний шар, і проходять через всі шари нейронної мережі до вихідного шару, де обчислюється функція помилки. Така операція виконується декілька разів під час навчання нейронної мережі. Будову нейронної мережі зображено на рисунку 2.1. На ньому вузлами позначено нейронами, а стрілками – з’єднання між нейронами і напрямок у якому передається інформація.

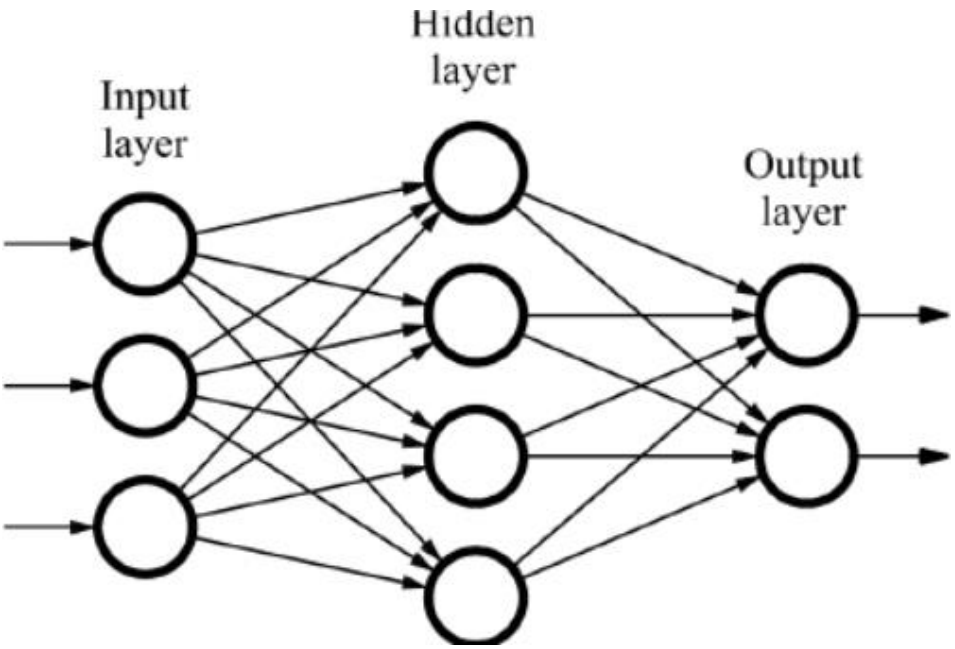


Рисунок 2.1 – Проста нейронна мережа

Розглянемо ШНМ як математичну модель, що визначає функцію f : X → Y , або розподіл над X , або над X та Y . Іноді моделі тісно пов'язують з певним правилом навчання.

З математичної точки зору, нейромережеву функцію $f(x)$визначають як композицію інших функцій $g_i(x)$ які може бути розкладено далі на інші функції. Широко вживаним способом компонування є нелінійна зважена сума:

$$f(x) = \mathrm{K}\sum_i w_i g_i f(x) \tag{2.1}$$

де K (що часто називають функцією збудження) є визначеною наперед функцією.

Найбільш розпоширеними є сигмоїдна функція та ReLU-функція.

Нейронна мережа складається з штучних нейронів — вузол штучної нейронної мережі, що є спрощеною моделлю природного нейрона. Математично, штучний нейрон зазвичай представляють як деяку нелінійну функцію від єдиного аргументу — лінійної комбінації всіх вхідних сигналів. Цю функцію називають функцією активації або функцією спрацьовування, передавальною функцією. Отриманий результат посилається на єдиний вихід. Такі штучні

нейрони об'єднують в мережі — з'єднують виходи одних нейронів з входами інших.Може приймати значення від одного до нуля, для цього використовуються передавальні функції. Функція активації, або передавальна функція штучного нейрона(рис. 2.2) — залежність вихідного сигналу штучного нейрона від вхідного. Зазвичай передавальна функція відображає дійсні числа на відрізку [0,1].Більшість видів нейронних мереж для функції активації використовують сигмоїди, але зараз все більше нейронних мереж використовують ReLU-функцію. Розглянемо ці дві активаційні функції у таблиці 2.1, а також декілька інших і визначимо яку буде продуктивніше використовувати під час розробки.

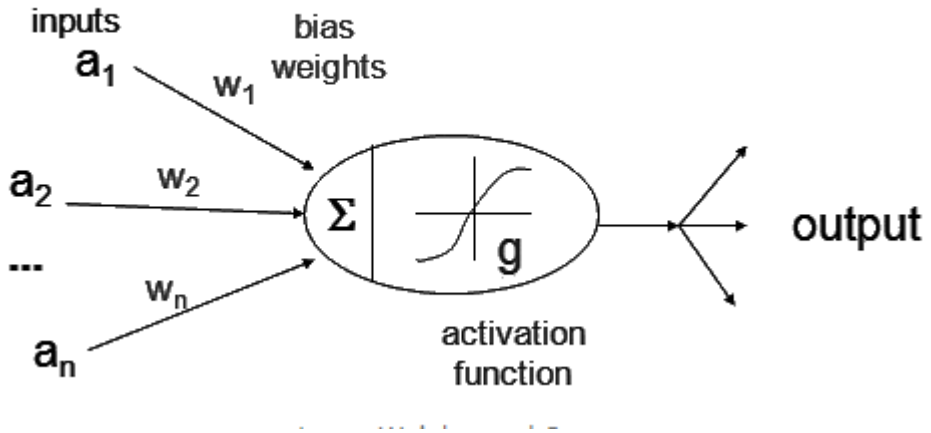
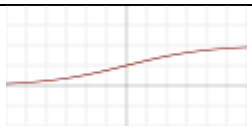
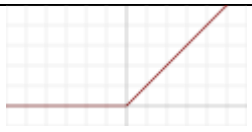


Рисунок 2.2 – Активаційна функція у нейронній мережі

Таблиця 2.1 – Порівняння сигмоїди і ReLU

Сигмоїда		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$
ReLU		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geqslant 0 \end{cases}$

Переваги ReLU порівнянно с сигмоїдою:

- ReLU набагато простіше обчислювати. Перехід вперед і назад через ReLU – просте if твердження, тобто процес буде відбуватися

набагато швидше. Швидкість малопомітна при малому розмірі вибірки, але стає значною з ії збільшенням .

- Сигмоїдні активації легше насичуються. Існує порівняно вузький інтервал входів, для якого похідна сигмоїда є достатньо ненульовою. Іншими словами, як тільки сигмоїда досягає лівого чи правого краю шару, робити безперервний прохід через неї нічого не дасть, оскільки похідна дуже близька до 0.
- ReLU істотно знижує обчислювальні витрати на навчання мережі. Це дозволяє навчати більші мережі з більшою кількістю параметрів при однакових обчислювальних витратах, що призводить до підвищення потужності, а також часто і більшої точності.

Алгоритм навчання нейронної мережі:

Всі нейрони мають активаційну функцію, що визначає значееня на виході нейрона. Цю функцію мі використовуємо для того щоб створити нелінійність під час моделювання мережі, активаційних функцій, як було сказано раніше, існує декілька і вони були розглянуті.

Тобто навчанням нейронної мережі ми можемо назвати процес уточнення параметрів w_{ij} i b_i – ваги і нейрон зміщення. Тобто ми можемо розглядати наш процес навчання, як ітераційний процес проходження нейронної мережі через всі шари нейронів вперед - forward propagation і назад -backpropagation.

Перший крок forwardpropagation, починається з вхідного шару, коли наша нейронна мережа має уже всі тренувальні дані, необхідні для прогнозування. Тобто на кожному кроці цієї ітерації нейрони застосовують дані обчислені нейронами з попереднього шару, використовуючи своє значееня від 0 до 1 і значення ваги і нейрону зміщення. Після того як дані перетнуть усі рівні нейронної мережі, при цьому проводячі розрахунки на кожному шарі, буде досягнуто шару виходу.

Далі ми будемо використовувати функцію втрат, щоб оцінити втрати (або помилку) та порівняти і виміряти, наскільки точний був наш прогноз по відношенню до правильного результату. В ідеалі ми хочемо, щоб наша функція втрат була нульовою, тобто без розбіжностей між оціночною та очікуваною вартістю. Тому, коли модель тренується, ваги взаємозв'язків нейронів будуть поступово коригуватися, поки не будуть отримані хороші прогнози.

Після обчислення втрат ця інформація поширюється назад. Звідси і його назва: розмноження. Починаючи з вихідного шару, інформація про втрати поширюється на всі нейрони в прихованому шарі, які безпосередньо сприяють виводу. Однак нейрони прихованого шару отримують лише частку загального сигналу втрати, виходячи з відносного внеску, який кожен нейрон вніс у вихідний шар. Цей процес повторюється, пошарово, до тих пір, поки всі нейрони мережі не отримають сигнал втрати, який описує їх відносний внесок у загальну втрату.

Візуально ми можемо підсумувати те, що ми пояснили цією візуальною схемою етапів на рисунку 2.3:

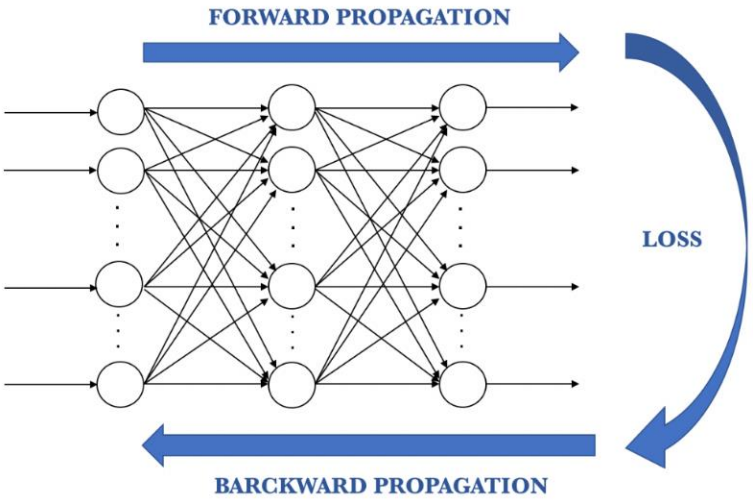


Рисунок 2.3 – алгоритм навчання нейронної мережі

2.1.1 Моделі нейронних мереж

1) CNN

Як і інші види штучних нейронних мереж, згорткова нейронна мережа має вхідний шар, вихідний шар та різні приховані шари. Деякі з цих шарів є складними, вони використовують свою власну математичну модель для передачі результатів послідовним шарам. CNN - це фундаментальний приклад глибокого навчання, де більш досконала модель підштовхує еволюцію штучного інтелекту, пропонуючи системи, що імітують різні види біологічної діяльності мозку людини.

Розберемо архітектуру згорткової нейронної мережі(Рис. 2.4).

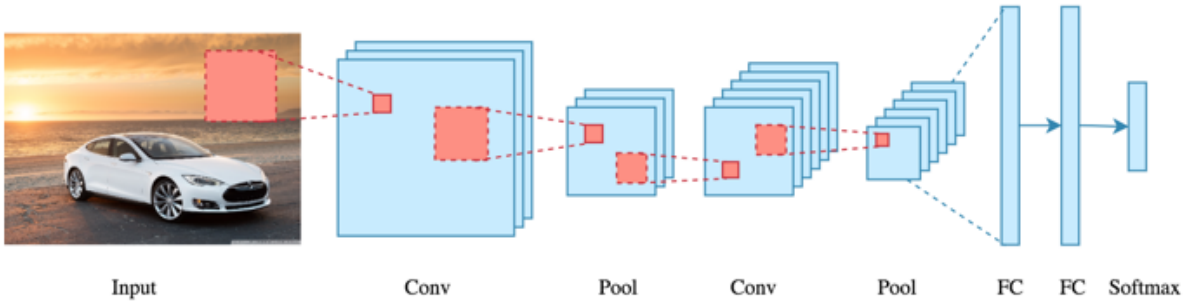


Рисунок 2.4 – Архітектура CNN

На вхід подається зображення з нашої вибірки. Далі виконується послідовність convolution + pooling (Згортка + пулінг) операцій. Потім вихідні дані передаються на шари зв’язаних між собою нейронів, тобто архітектура така ж як у ШНМ. Тепер більш докладно розглянемо кожен елемент.

1) Згортка

Математично операція згортки - це підсумок поелементного добутку двох матриць. Візьмемо дві матриці, X і Y. Якщо нам потрібно "згорнути зображення X за допомогою фільтра Y", ця операція створить матрицю Z. Потім треба знайти суму всіх елементів матриці Z, отримане скалярне значення і буде результатом згортки. Суть згортки зображено на рисунку 2.5.

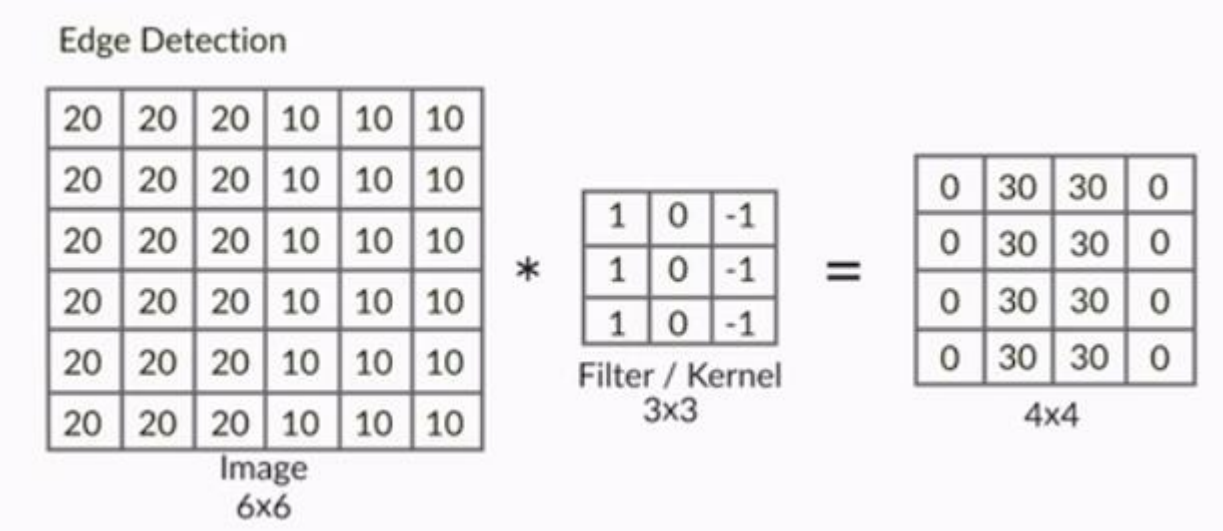


Рисунок 2.5 – Операція згортки зображення 6×6 з фільтром 3×3

2) Субдискретизація

Шар субдискретизації розглядає більші регіони зображення і фіксує їх сукупну статистику (максимальне значення, середнє значення тощо) кожної області. Зазвичай 2×2 піксель перетворюється у один. Зобразимо алгоритм субдискретизації на рисунку 2.6.

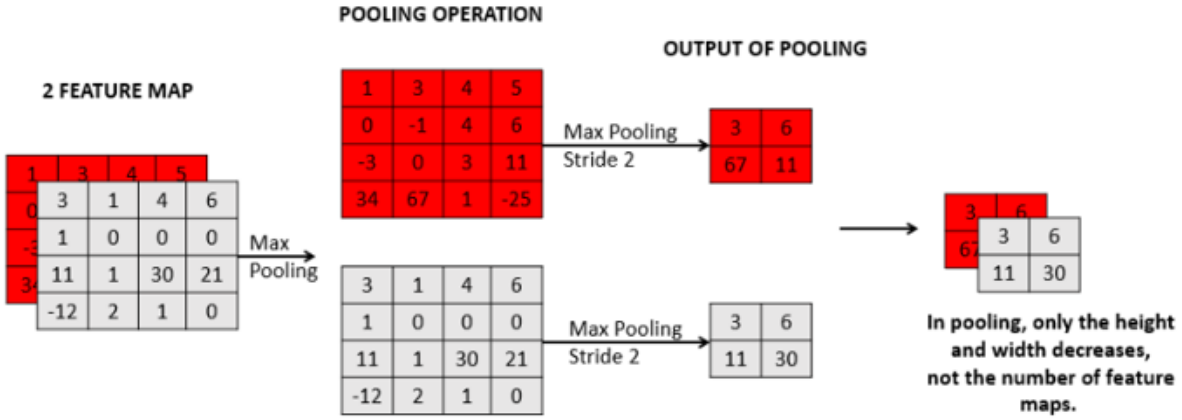


Рисунок 2.6 – субдискретизація

Дві найпопулярніші функції, які використовуються при пулінгу – максимальне і середнє значення. Максимальне – беруться прямокутники які не перетинаються між собою, і з кожного вибирається максимальне значення, яке передається у наступний шар. Аналогічною є операція для знаходження середнього.

3) Нейронна мережа

Після декількох шарів згортки і дискретизації дані об'єднуються і передаються на звичайну повнозв'язну нейронну мережу, яка теж може складатися з декількох шарів.

Далі розглянемо регуляризацію згорткової нейронної мережі. Для кращої роботи моделі використовується дуже поширена методика регуляризації, тобто додавання вибраної функції регуляризації до цільової функції. Це здійснюється насамперед для запобігання перетренування моделі. Найпоширенішим видом цього є dropout.

Операція dropout (випадання) – під час проходження кожного шару нейронної мережі, деякі нейрони стають неактивними, тобто його викидають. Ймовірність випадання нейрону на поточному кроці визначає параметр p. Це означає що нейрон буде неактивний на вході і на виході на поточній ітерації, але може бути знову активований у наступних. Параметр p називається шансом випадання, і зазвичай має значення 40-50%. Dropout може значно підвищити ефективність згорткової нейронної мережі, навіть якщо до найкращих нейронних мереж з точністю > 90% додати операцію випадання, їх результативність зросте на 2-3%, що являється дуже значним покращенням на такій точності. Алгоритм виконання dropout зображено на рисунку 2.7.

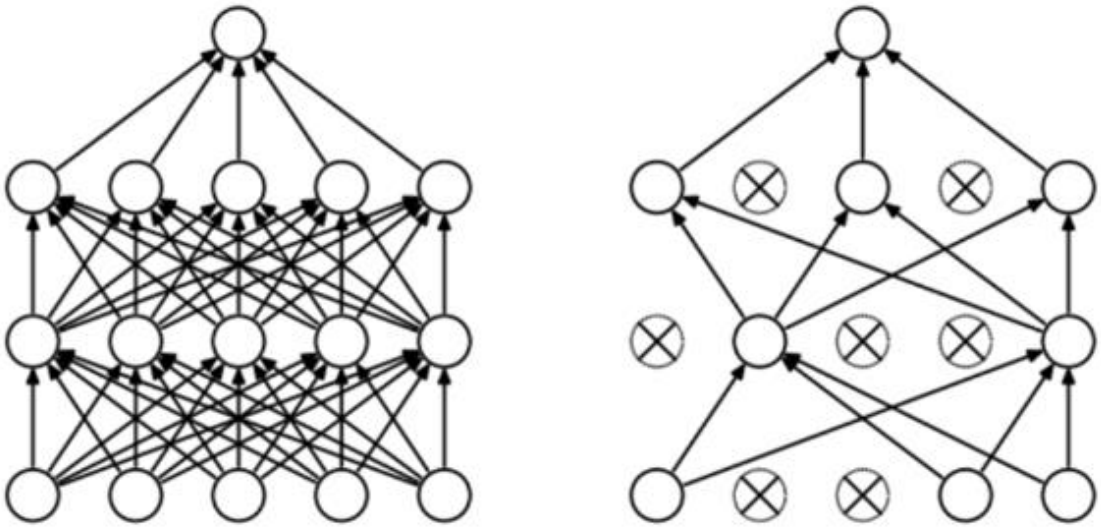


Рисунок 2.7 – Dropout

Переваги CNN:

- Можливість навчання на графічних процесорах, що значно підвищує швидкість навчання нейронної мережі
- Дуже добре підходить для задачі розпізнавання зображень, що може бути використано для ЕКГ.
- Стійкість до здвигів і різних шумів на зображеннях.
- Застосування градієнтного спуску під час навчання.

До недоліків можна віднести занадто велику кількість параметрів. Тобто ми точно не знаємо які параметри краще використовувати для розв’язання різних типів задач.

2) RNN

Рекурентна нейронна мережа (RNN), належить до класу штучних нейронних мереж, де зв’язки між нейронами утворюють спрямований цикл. Це створює внутрішній стан мережі, що дозволяє їй проявляти динамічну поведінку у часі. RNN можуть використовувати свою внутрішню пам'ять для обробки довільних послідовностей входів. У RNN сигнали рухаються як вперед, так і назад, створюючі петлі в мережі. Різницю між рекурентною нейронною мережею і звичайно. Можна побачити на рисунку 2.8.

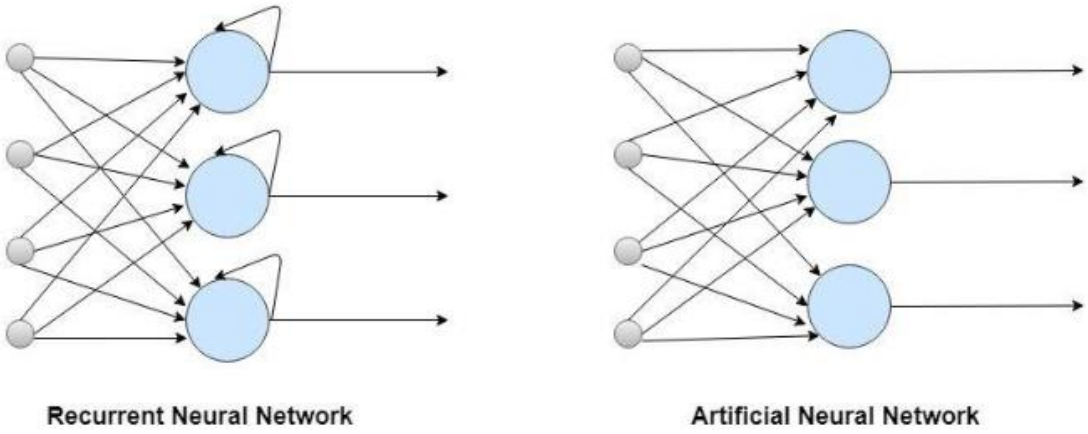


Рисунок 2.8 – Різниця між рекурентною нейронною мережею і звичайною нейронною мережею

У цій нейронній мережі вихідні дані з попереднього шару передаються на вхід до поточного. Звичайні нейронні мережі, що мали місце до розвитку рекурентних нейронних мереж, всі входи та виходи незалежні один від одного. За приклад можемо взяти розпізнавання тексту - коли необхідно передбачити наступне слово речення, нам потрібно знати попередні слова і таким чином важливо їх запам’ятовувати. RNN вирішує цю проблему за допомогою прихованого шару. Ключовою і найбільш важливою особливістю рекурентної нейронної мережі є наявність прихованого стан, який запам'ятовує деякі деталі з попередніх ітерацій. RNN має пам'ять, яка зберігає всі обчислені дані. Також RNN використовує однакові параметри на всіх кроках, адже на всіх входах і прихованих шарах використовується однакова функція. Такий підхід значно зменшує складність задання і обчислення параметрів, а отже і пришвидшує час навчання.

У звичайних нейронних мережах на кожному шарі є свої набори b і w – нейронів зміщення і ваги, які різняться на кожному шарі. При чому вони не мають жодної інформації про те, що відбувалося на попередньому шарі. Тоді як рекурентні нейронні мережі працюють за наступним принципом: Спочатку RNN здійснює перетворення незалежних один від одного активацій в залежні активації. Також призначається однакова вага та значення нейрону зміщення всім шарам, що ще більше зменшує складність параметрів RNN та забезпечує послідовну основу для запам'ятовування попередніх вихідних даних. Тоді декілька шарів з однаковими b і w зливаються в один рекурентний шар.

Формула для визначення поточного стану:

$$h_t = f(h_{t-1}, x_t) \tag{2.2}$$

де $h_t -$ поточний стан,

$h_{t-1} -$ попередній стан,

$x_t -$ стан на вході.

Формула для активаційної функції:

$$h_t = \tan(W_{hh}h_{t-1}, W_{xh}x_t) \tag{2.3}$$

де W_{hh} – значення ваги на вхідному шарі,

W_{xh} – значення ваги на рекурентному шарі

Для визначення вихідних даних:

$$y_t = W_{hy}h_t \tag{2.4}$$

Одним із головних недоліків стандартних RNN є проблема зникаючого градієнта при якій продуктивність нейронної мережі спадає, оскільки вона не може бути навчена належним чином. Проблема зникаючого градієнта проявляється лише у масивних RNN, які навчаються на великій кількості даних. Якщо RNN використовує стандартний метод градієнтного спуску, то чим вона більша тим гіршу точність вона буде мати.

Найпоширенішим варіантом вирішення проблеми зникаючого градієнта являється LSTM - Long short-term memory, який зображено на рисунку 2.9. Це різновид архітектури RNN в якій інформація аналізується і зберігається в блоки довгострокової та короткострокової пам’яті. Це дає змогу RNN розібратися, які дані є важливими для запам’ятовування і зберігання, а які можна відкинути.

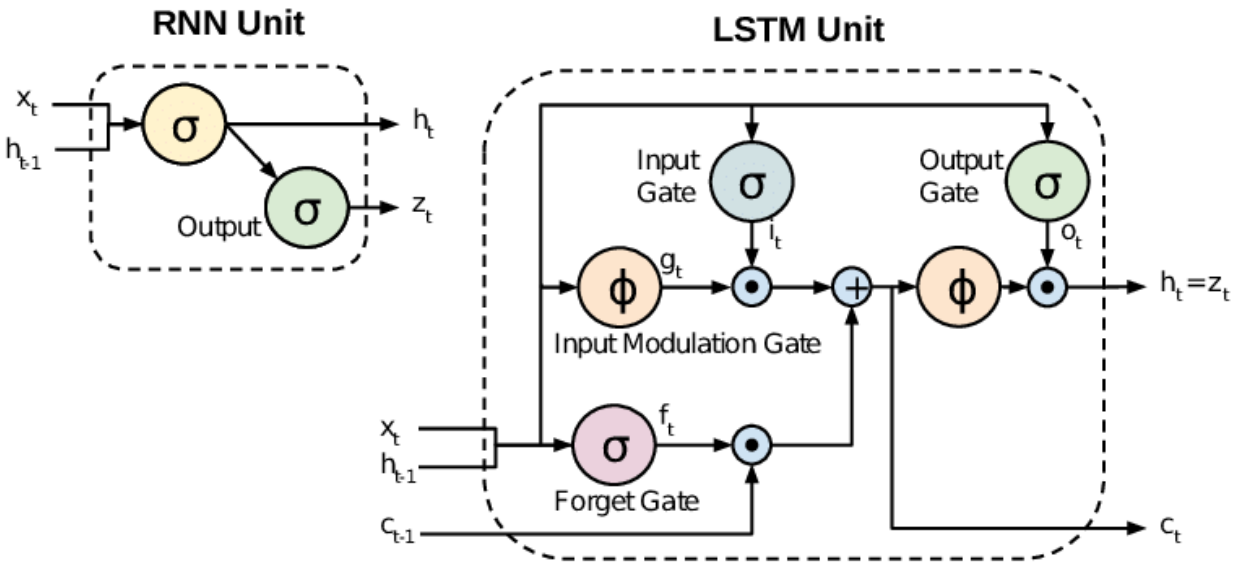


Рисунок 2.9 – LSTM Unit

Переваги RNN:

- Може обробляти вхідні дані будь якого об’єму.
- Зберігає важливу і відкидає неважливу інформацію на кожній ітерації під час тренування.
- Однакові вагові коефіцієнти на кожній ітерації.

Недоліки RNN:

- Через свою рекурсивність, досить повільно навчається.
- Зникаючий градієнт
- Навчання а також тестування різних проблем можуть бути достатньо складними

2.2 Методи обробки часових сигналів

2.2.1. Фрактальний метод

Фрактал - це самоподібна підмножина евклідового простору, фрактальний вимір якого суворо перевищує його топологічний вимір. Фрактали виявляються однаковими на різних рівнях, що показано в послідовних збільшеннях набору Мандельброта, через це фрактали зустрічаються всюди в природі. Фрактали демонструють подібні зразки на все більш малих масштабах, які називаються самоподібністю, також відомими як розширення симетрії або розгортання симетрії. Якщо ця реплікація абсолютно

однакова в усіх масштабах, вона називається афінною самоподібною. Фрактальна геометрія лежить в межах математичної галузі теорії мір.

Одною з головних властивостей фрактала вважається самоподібність - інваріантність щодо переносів і скейлінга (зміни масштабу). Це означає, що фрактал в будь-якій мірі одноманітно влаштований у великому діапазоні масштабів. Наприклад, якщо збільшувати маленькі елементи фрактала, то вони вийдуть схожими на великі. В ідеальному випадку фрактальний об'єкт є інваріативним щодо розтягувань, іншими словами йому властива незмінність при зміні масштабу найбільш істотних геометричних особливостей фрактала. У більшості природних об'єктів спостерігається самоподібність, яка є основним організуючим принципом фракталів. Тому не залежно від використовуваної шкали, фрактали будуть нагадувати один одного.

Аналітично фрактали зазвичай ніде не диференціюються. Нескінченна фрактальна крива може розглядатися як звивиста через простір інакше, ніж звичайна лінія - хоча вона все ще є одновимірною, її фрактальна розмірність означає, що вона також нагадує поверхню.

Зазвичай фрактали ділять на алгебраїчні, геометричні та стохастичні. Стохастичні фрактали при певних умовах можуть називатися мультифрактали. Також існують і інші класифікації: - Рукотворні і природні(Рис 2.4). Рукотворні - фрактали, при будь-якому масштабі володіють фрактальними властивостями.

Природні - фрактали з обмеженням області існування - максимальний і мінімальний розмір, при яких фрактальні властивості спостерігаються у об'єкта.

1) Алгебраїчні фрактали

Для побудови фракталів даного типу застосовуються ітерації нелінійних відображень, задані алгебраїчними формулами. Побудова фракталів здійснюється двома способами: перший - використання L-систем (від імені Lindenmayer), другий - використання системи IFS (iterated function systems). L-

система являє собою граматику деякого мови, що описує ініціатор і перетворення, яке над ним виконується за допомогою засобів, схожих із засобами мови Лого.

Побудова IFS - фракталів здійснюється за наступним алгоритмом: вибирається деякий безліч правил здійснення переходу з поточної точки в наступну, з імовірністю p застосування кожного правила. Починаючи з точки (0; 0), правило вибирається випадковим чином і визначається наступна точка, з неї знаходиться наступна, і так далі. Побудова зображена на рисунку 2.10.

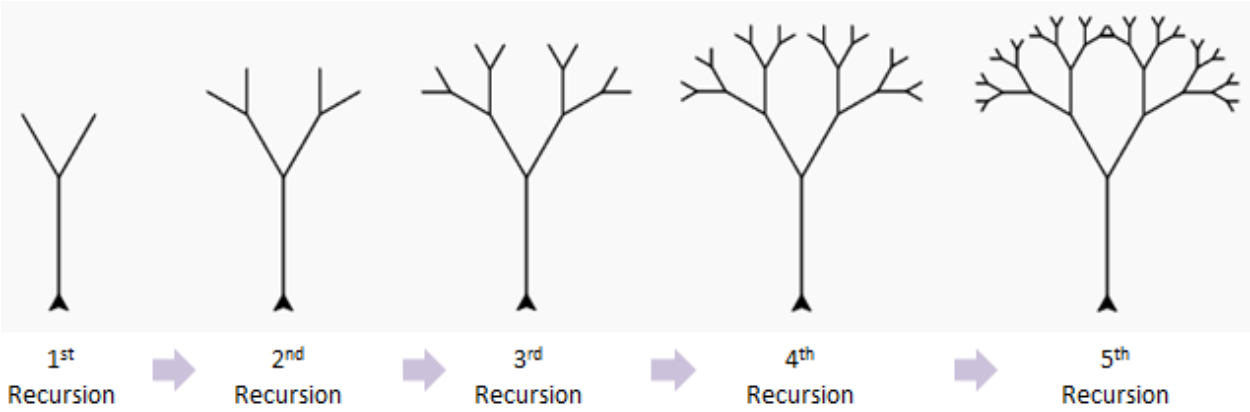


Рисунок 2.10 – Побудова фрактального дерева

2) Геометричні фрактали(Рис 2.11)

В даних фракталах відразу видно самоподібність. У двовимірному випадку вони задають деяку ламану, яка називається генератором, і за один крок алгоритму кожен відрізок ламаної змінюється на ламану-генератор, причому у відповідному масштабі. Після багаторазового повторення даної процедури утворюється фрактальная крива.

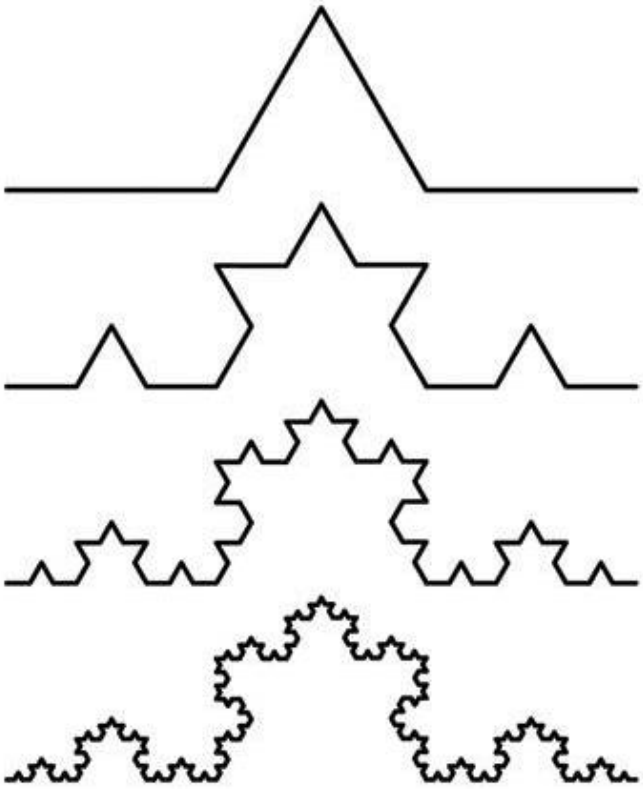


Рисунок 2.11 – Крива Коха

3) Стохастичні фрактали

Фрактали, при побудові яких змінюються різні параметри абсолютно випадковим чином, називаються стохастичними. Вони знаходять своє відображення в фізичних процесах, тому вони найбільш цікаві для фізиків. Двовимірні стохастичні фрактали застосовуються при моделюванні поверхні моря або рельєфу місцевості.

Якщо ми розглядаємо електрокардіографічний сигнал в часі, то він має властивості фрактала (беремо ми короткі або тривалі ділянки записи кривої, вони мають певну схожість між собою).

При описі фрактала однією з найбільш важливих його характеристик є фрактальна розмірність:

$$D = 2\text{-}H \tag{2.5}$$

де H - показник Херста.

Разом з нею (фрактальної розмірністю), через показник Херста часто обчислюють і інші фрактальні параметри:

Кореляційний параметр β який вираховується за формулою:

$$\beta = 2\,(1\text{-}H) \tag{2.6}$$

Спектральний показник b обчислюється за формулою:

$$b = 2H + 1 \tag{2.7}$$

Фрактальний показник α обчислюється за формулою:

$$\alpha = 3\text{-}2H \tag{2.8}$$

Як випливає з наведених формул, одним з ключових показників характеризують фрактал є параметр Херста. У фізичному сенсі параметр Херста характеризує ступінь порізаності (звивистості) досліджуваного графіка. Параметр Херста приймає значення в діапазоні від 0 до 1 ($0 < H < 1$). При малих значеннях параметра Херста графік функції має велику порізаність, при великих значеннях - порізаність носить більш плавний характер.

Отже ми можемо розглядати сигнал ЕКГ як фрактал. Основними показниками на які слід звернути уваги є фрактальна розмірність, показник Херста і спектральний показник.

Розглянемо методи визначення фрактальної розмірності сигналу.

1) Метод Каца

FD кривої сигналу, отриманої на методі Каца можна визначити як:

$$\mathrm{FD} = \frac{\log(L)}{\log(D)} \tag{2.9}$$

де L - загальна довжина кривої сигналу або сума відстані між послідовними точками;

D - оцінка діаметра відстані між першою точкою та даними, що дає найбільшу відстань.

D і L, відповідно, можна виразити, як:

$$D = \max ||X_i - X_1|| \tag{2.10}$$

$$L = \sum_{i=1}^{N} ||X_{i+1} - X_1|| \tag{2.11}$$

Нормалізуючи відстані в (2.9) на середню відстань між послідовними точками, на a , дає:

$$\text{FD} = \frac{\log(\frac{L}{a})}{\log(\frac{d}{a})} \tag{2.12}$$

Якщо визначити n як кількість кроків сигнальної кривої , де n < N, $n = \frac{L}{a}$. Покажемо фрактальні розмірності різних захворювань і здорового серця знайдених методом Каца у таблиці 2.2.

Таблиця 2.2 – Значення фрактальної розмірності різних ЕКГ сигналів знайдених методом Каца

Сигнал	Фрактальна розмірність
Серцевий блок	1.345 - 1.3733
Фібриляція передсердь	1.1013- 1.5564
Надшлуночкова аритмія	1.2762 - 1.5698

Нормальний серцевий ритм	1.114 -1.6051

2) Метод Хігучі

Хігучі запропонував ефективний алгоритм для обчислення ФР безпосередньо з часових рядів. Припустимо одновимірний часовий ряд X = {X (1), X (2), X (3), ..., X (N)} де, N - загальна кількість вибірок, у нашому випадку серія X - послідовні значення сигналу ЕКГ. Алгоритм Хігучі будує k нових часових рядів:

$$X_k^m = \{X(m), X(m+k), \dots, X(m+Mk)\} \tag{2.13}$$

де k і m - цілі числа, які представляють собою часовий інтервал між точками і початковим значенням часу відповідно,

$M = \frac{N-m}{k}$ Для кожного нового побудованого часового ряду X_k^m довжина L_k^m обчислюється як:

$$L_k^m = \sum_{i=1}^{M} \frac{N-1}{MK} X(m+ik) - X(m+(i-1)k)\} \tag{2.14}$$

де $\frac{N-1}{MK}$ - коефіцієнт нормалізації довжини кривої X_k^m.

Довжина ряду L(k) для часового інтервалу k обчислюється як середнє значення k, для m = 1, 2, ..., k:

$$L(k) = \frac{1}{k} \sum_{m=1}^{k} L_k^m \tag{2.15}$$

Якщо L (k) пропорційна k^{-FD}, то крива, що описує форму часового ряду ЕКГ, є фрактальною з розмірністю FD. У цьому випадку, якщо ln(L(k))

побудувати разом з ln(k), k = 1, 2, 3, ...,K_{max} точки падають на пряму з нахилом, рівним FD. Фрактальна розмірність ЕКГ-сигналу обчислюється за вищевказаним методом при застосуванні адаптивного та фіксованого методу вікон. Покажемо фрактальні розмірності різних захворювань і здорового серця знайдених методом Хігучі у таблиці 2.3.

Таблиця 2.3 – Значення фрактальної розмірності різних ЕКГ сигналів знайдених методом Хігучі

Сигнал	Фрактальна розмірність
Серцевий блок	1.8081 - 1.8155
Фібриляція передсердь	1.5401- 1.9910
Надшлуночкова аритмія	1.7745 - 1.9881
Нормальний серцевий ритм	1.3996 -1.9771

3) Метод зміни розміру (R / S)

Херст розробив метод R / S, який є статистичною методикою для аналізу великої кількості природних явищ . Метод R/S - один з найдавніших і найвідоміших методів оцінки H (параметр Херста). Нехай $\{X_k\}$, $k = 1, 2, 3, \ldots, N$ - набір N вибіркових точок записів ЕКГ. Середнє X(N) і стандартне відхилення S(N) цих точок відповідно становлять:

$$X(N) = \sum_{i=1}^{N} X_i \tag{2.16}$$

$$S(N) = \sum_{i=1}^{N} (X_i - X(N))^2 \tag{2.17}$$

а R / S-статистика або відрегульований коригований діапазон, визначається співвідношенням:

$$\frac{R(N)}{S(N)} = \frac{\max\{0, w_i\}_{i=1}^{N} - \min\{0, w_i\}_{i=1}^{N}}{S(N)} \quad (2.18)$$

де $w_k = (X_1 + X_2 + \cdots + X_k) - KX(N), k = 1,2,3, \ldots, N$

Херст емпірично встановив, що вони представлені співвідношенням

$$\frac{R(N)}{S(N)} \approx CN^H, \quad N \to \infty \quad (2.19)$$

де C - кінцева позитивна константа.

Підставивши у логарифм, ми отримуємо:

$$\log\left(\frac{R(N)}{S(N)}\right) \approx \log(C) + Hlog(N) \quad (2.20)$$

Співвідношення між показником Херста і фрактальним виміром просто визначається як FD = 2-H. Тож фрактальну розмірність за допомогою цих рівнянь легко оцінити в аналізі масштабного діапазону. Покажемо фрактальні розмірності різних захворювань і здорового серця знайдених методом R/S у таблиці 2.4.

Таблиця 2.4 – Значення фрактальної розмірності різних ЕКГ сигналів знайдених методом R/S

Сигнал	Фрактальна розмірність
Серцевий блок	1.3723 - 1.3697
Фібриляція передсердь	1.0739 - 1.2441
Надшлуночкова аритмія	1.2152 - 1.3123
Нормальний серцевий ритм	1.3377 - 1.3634

4) Метод спектру потужності(PSM)

Ми можемо застосовувати фрактальний метод, коли основний процес, який моделюється математично, має подібний вигляд незалежно від масштабу, над яким він спостерігається. Виявляється, багато природних сигналів можна змоделювати за допомогою фракталів. Багато сигналів, які спостерігаються в природі, є випадковими фракталами, включаючи біомедичні сигнали, такі як сигнал часового ряду ЕКГ. Фрактальні сигнали випадкових масштабів (RSF) - це сигнали, функція розподілу ймовірностей яких (PDF) має однакову "форму" незалежно від масштабу, над яким вони спостерігаються. Так що випадкові фрактальні сигнали статистично є подібними (самоподібність). Сигнали RSF характеризуються спектрами потужності, розподіл частоти яких пропорційний $\frac{1}{f^q}$ де f - частота, а q > 0 - "розмірність Фур'є", значення, яке просто пов'язане з параметром FD та H, за співвідношенням:

$$q = H + 1/2 = (5 - 2D)/2 \tag{2.21}$$

Цей закон потужності описує звичайні моделі RSF, які базуються на стаціонарних процесах, в яких «статистика» сигналів RSF інваріантна за часом і значення q є константою.

Припустимо що X(t), у часовій області, є часовим рядом сигналу ЕКГ, який вважається сигналом самоафінності. Зауважте, що на кожному з малюнків 6 і на рисунку 7 показано нормальний ЕКГ-сигнал та сигнал з дефектами. Спектр потужності такого сигналу можна записати як:

$$P(f) = |X(f)|^2 \tag{2.22}$$

де $X(f)$- швидке перетворення Фур'є часового ряду в частотній області.

Для таких часових рядів спектр потужності P(f):

$$P(f) = \frac{c}{f^q} \tag{2.23}$$

Нехай $X_2, X_3, \dots, X_N$ - вибіркові точки сигналу ЕКГ. Розглянемо випадок, коли цифровий спектр потужності $P_i = P(f_i)$ задають, застосовуючи перетворення Фур'є до цього часового ряду:

$$F(f_i) = \frac{c}{|f_i|^{\frac{q}{2}}} \tag{2.24}$$

Звідки маємо

$$q = \frac{N\sum_{i=1}^{N}(lnf_i)lnP(f_i) - \sum_{i=1}^{N}(lnf_i)\sum_{i=1}^{N}lnP(f_i))}{\sum_{i=1}^{N}(lnf_i)^2 - N\sum_{i=1}^{N}(lnf_i)} \tag{2.25}$$

Зобразимо нормальний серцевий ритм і надшлуночквою аритмію для порівняння на рисунках 2.12 і 2.13.

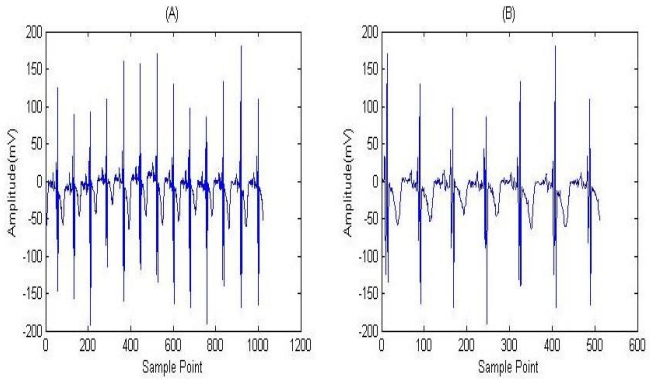


Рисунок 2.12 – (зліва) Нормальний серцевий ритм 1024 точки,(зправа) наближений графік нормального серцевого ритму 512 точок

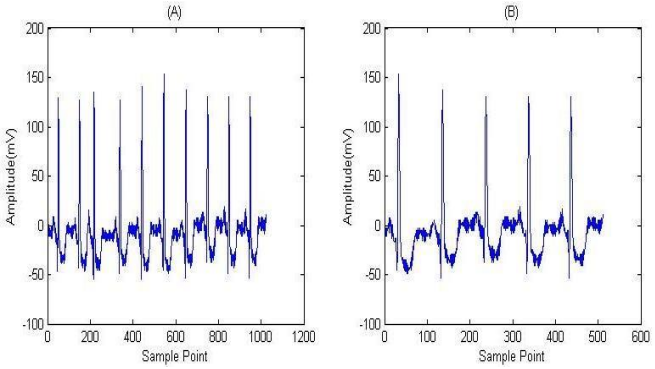


Рисунок 2.13 – (зліва) Надшлуночкова аритмія, 1024 точки,(зправа) наближений графік надшлуночкової аритмії, 512 точок

Покажемо фрактальні ррозмірності різних захворювань і здорового серця знайдених методом PSM у таблиці 2.5.

Таблиця 2.5 – Значення фрактальної розмірності різних ЕКГ сигналів знайдених методом PSM

Сигнал	Фрактальна розмірність
Серцевий блок	0.0859 - 0.1229
Фібриляція передсердь	0.1401 - 0.3795
Надшлуночкова аритмія	0.3806 – 0.5165
Нормальний серцевий ритм	1.1617 - 1.7882

Порівняємо середні значення фрактальної розмірності, знайденої різними методами

Таблиця 2.5 – Порівняльна таблиця с середніми значеннями фрактальної розмірності

Сигнал	Метод Каца	Метод Хігучі	RS метод	PSM
Серцевий блок	1.7918	1.3351	1.3627	0.1161

Фібриляція передсердь	1.7764	1.3434	1.7922	0.2806
Надшлуночкова аритмія	1.8304	1.4066	1.2757	0.4314
Нормальний серцевий ритм	1.7931	1.3859	1.3483	1.5156

Отже можемо зробити висновок, що PSM метод показує більшу різницю між здоровими і хворими людьми, а отже є найточнішим.

2.2.2 Перетворення Фур’є

У ряд Фур’є можуть бути розкладені як безперервні так і дискретні сигнали. У нашому випадку доцільно розглядати лише дискретний випадок, оскільки сигнал ЕКГ являється саме дискретним сигналом. Тобто будь-яку періодичну функцію мі можемо представити у вигляді суми синусів і косинусів, а також ми можемо обчислити спектр цього сигналу. Дискретне перетворення Фур'є має вигляд:

$$X_k = \frac{1}{N}\sum_{i=0}^{N-1} x_n e^{-j\frac{2\pi ki}{N}} = \frac{1}{N}\sum_{i=0}^{N-1} x_n [cos\frac{2\pi ki}{N} - jsin\frac{2\pi ki}{N}] \quad (2.26)$$

Формула Ейлера – зв’язує комплексну експоненту і тригонометричні функції:

$$e^{-j\varphi} = cos\varphi + jsin\varphi \quad (2.27)$$

Після цього перетворення сигнал буде представлений у вигляді коефіцієнтів, які відповідають амплітудам і фазам частот, складаючих цей сигнал.

$$ReX_k = \frac{1}{N}\sum_{i=0}^{N-1} cos\frac{2\pi ki}{N} \tag{2.28}$$

$$ImX_k = -\frac{1}{N}\sum_{i=0}^{N-1} sin\frac{2\pi ki}{N} \tag{2.29}$$

$$\varphi[X_k] = tan^{-1}\frac{ImX_k}{ReX_k} \tag{2.30}$$

Отже якщо ми маємо вхідний сигнал з N елементів, перетворення Фур'є розкладає його на $N\ /\ 2 + 1$ синусних та $N\ /\ 2 + 1$ косинусних частини. Тобто розкладає функцію в свій спектр – кінцеве число частот з певними амплітудами і фазами.

Зворотне дискретне перетворення Фур'є має вигляд:

$$x_n = \frac{1}{N}\sum_{i=0}^{N-1} X_k e^{-j\frac{2\pi ki}{N}} = \frac{1}{N}\sum_{i=0}^{N-1} X_k[cos\frac{2\pi ki}{N} - jsin\frac{2\pi ki}{N}] \tag{2.31}$$

Використовується для задачі обчислення вхідного сигналу з відомими частотними складовими.

Перетворення Фур'є використане для обробки вхідного сигналу може визначити, які частоти є у нашому сигналі та в яких пропорціях вони існують.

Основні причини використання перетворення Фур'є при обробці сигналів:

- $|X(f)|^2$, показує потужність сигналу $x(t)$ на певній частоті f.
- З теореми Персеваля маємо:

$$\int_R |x(t)^2|dt = \int_R |X(f)^2|df \tag{2.32}$$

- Тобто енергія за весь час сигналу, дорівнює сумі енергії на всіх частотах. Таким чином, трансформація зберігає енергію.
- Маючи змогу розділити сигнали на їх складові частоти, можна легко блокувати певні частоти вибірково
- Зміщений сигнал у часовій області проявляється фазовою зміною частотної області. Хоча це підпадає під категорію елементарних властивостей, це властивість, яка широко застосовується на практиці, особливо в програмах візуалізації та томографії
- Обчислення похідних сигналу.

2.3 Висновки до розділу

У другому розділі було розглянуто нейронні мережі, їхню будову і алгоритм навчання. Також було проаналізовано дві моделі нейронних мереж найбільш підходящих для системи обробки і класифікації ЕКГ – CNN і RNN. Також було розглянуто методи обробки сигналів, а саме фрактальний метод і перетворення Фур’є. Для фрактального було проаналізовано 4 варіанти знаходження фрактальної розмірності функції і визначено найточніший з них.

РОЗДІЛ 3 Архітектура та аналіз результатів роботи

3.1 Вибір мови і середовища для розробки програмного продукту

Розберемо існуючі мови програмування і виберемо найкращу для реалізації системи обробки і класифікації кардіограм.

3.1.1 Python

Якщо йде мова про машинне навчання і нейронні мережі, першим на думку приходить саме Python. Перш за все у Python наявна величезна кількість бібліотек для машинного навчання, які значно пришвидшують а також роблять роботу розробника простішою, так як можна застосовувати готові рішення. По друге, ця мова програмування має гарну читабельність і простий синтаксис роблять мову зрозумілою для тих, хто розробляв продукти но іншій мові програмування. Через це програмісти можуть не зациклюватись на синтаксисі мови, а фокусуватися на побудові моделі, аналітичній і математичній частині проекту. Також Python завжди розвивається, особливо у напрямі розробки штучного інтелекту, це означає що завжди будуть з’являтися нові рішення і бібліотеки для задач машинного навчання.

Розглянемо бібліотеки які використовуються під час навчання (рис 3.1):

1) NumPy - це дуже популярна бібліотека Python для обробки великих масивів і матриць, яка використовує математичні і статистичні функції високого рівня.

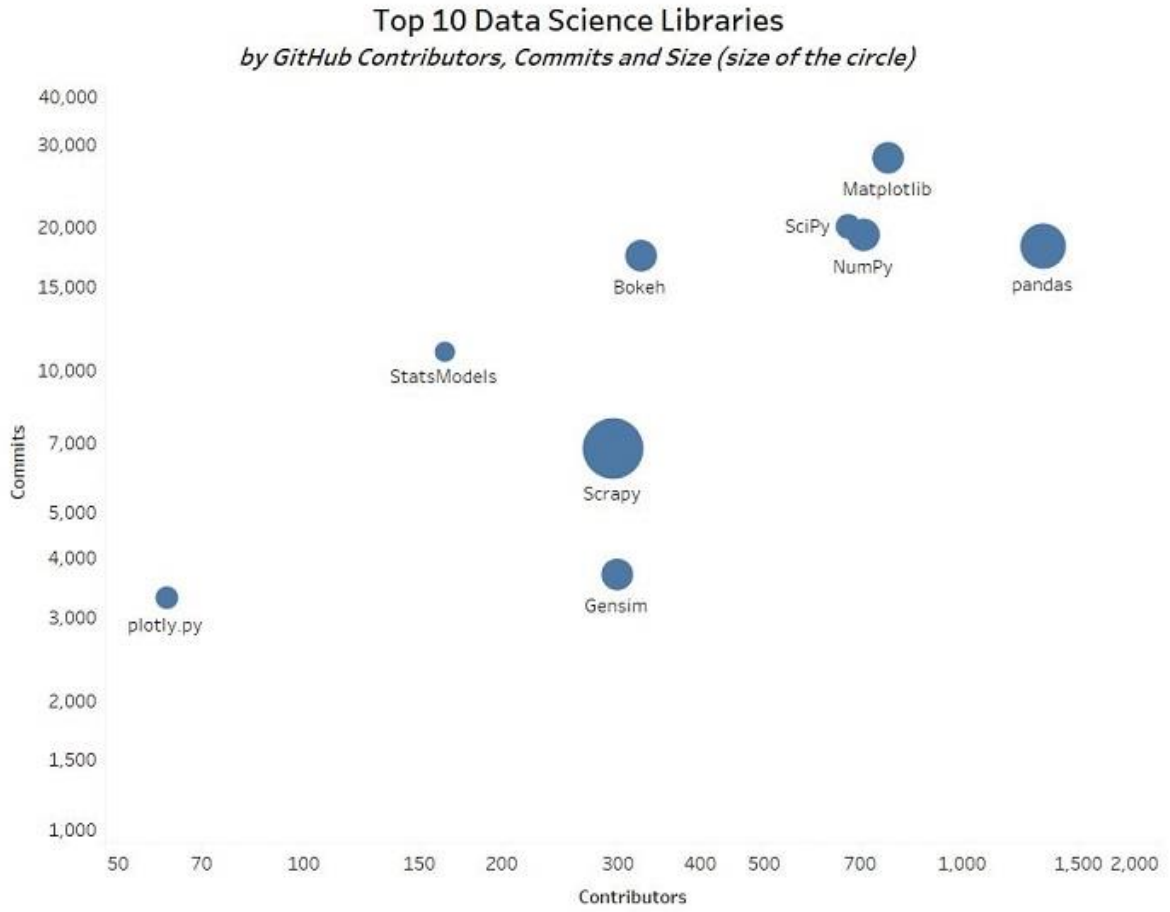


Рисунок 3.1 - Найпопулярніші бібліотеки для машинного навчання

2) SciPy - бібліотека, яка містить різні модулі для оптимізації, лінійної алгебри, інтеграції та статистики, вона також гарно проводить різні модифікації з зображеннями, що дуже корисно для збільшення кількості картинок в навчальному датасеті.
3) Skikit-learn - одна з найпопулярніших бібліотек для класичних алгоритмів ML. Вона побудована на основі двох основних бібліотек Python, а саме: NumPy та SciPy. Scikit-learn підтримує більшість контрольованих та неконтрольованих алгоритмів навчання. Scikit-learn також може бути використаний для пошуку даних та аналізу даних, що робить його чудовим інструментом для застосування у сфері машинного навчання.
4) TensorFlow - бібліотека з відкритим кодом для чисельних обчислень з високою продуктивністю, розроблена командою Google Brain в Google. Як

випливає з назви, Tensorflow - це рамка, яка включає визначення та запуск обчислень за участю тензорів. Вона може тренувати та запускати глибокі нейронні мережі. TensorFlow широко застосовується в галузі машинного навчання.

5) Pandas - Основна Python бібліотека для роботи з даними. Ця бібліотека не пов’язана на пряму з машинним навчанням, але для тренування і тестування нейронних мереж нам необхідно підготувати дані - очистити їх від шумів, структуравати у таблиці, перенести у правильні формати. Pandas надає структуру даних високого рівня та широкий спектр інструментів для аналізу даних, також надає багато вбудованих методів збирання, комбінування та фільтрації даних.
6) Matpoltlib - бібліотека Python для візуалізації даних. Так само як і Pandas напряму не пов’язана у навчанні нейронних мереж. Основне застосування Matpoltlib це візуалізація даних - побудова графіків у різних площинах, графів, діаграми, гістограми.
7) Statsmodels - це пакет Python, який описує наукові дослідження та статистичні обчисленя. Включає в себе описову статистику та оцінку для статистичних моделей
8) Keras -порівняно нова, але дуже швидко зростаюча бібліотека Python. Це API нейронних мереж високого рівня, здатна працювати на TensorFlow або Theano. Він може працювати безперебійно як на процесорі, так і на GPU. Keras робить створення моделей та нейронних мереж дуже простим і швидким процесом, оскільки має готові методи для створення різноманітних шарів і операцій з ними.

3.1.2 C++

C++ - об’єктно- орієнтована мова програмування високого рівня. Вважається достатньо складною для вивчення, але надає можливість самому контролювати багато аспектів своєї програми. Через це C++ має дуже високу

швидкість виконання коду, значно більше ніж у Python, але не має такої кількості бібліотек які застосовуються саме для машинного навчання.

Розглянемо основні бібліотеки для машинного навчання на мові програмування C++ :

1) Shark - це швидка модульна бібліотека C++, і вона має підтримку контрольованих алгоритмів навчання, таких як лінійна регресія, нейронні мережі, кластеризація, k-засоби тощо. Також вона включає функціональність лінійної алгебри та числову оптимізацію. Це ключові математичні функції або сфери, які дуже важливі при виконанні завдань машинного навчання.
2) mlpack - це швидка і гнучка бібліотека машинного навчання, написана на C ++. Вона спрямована на швидке та розширене впровадження передових алгоритмів машинного навчання. mlpack надає ці алгоритми як прості програми у вигляді командного рядка C ++, які потім можуть бути інтегровані у більш масштабні рішення машинного навчання.

3.1.3 Swift

Swift - нова мова програмування створена Apple, переважно для застосування для iOS і MacOS. Але через високу швидкість, простому синтаксису та новизні, багато людей прогнозують Swift майбутнім машинного навчання.

Хоча Swift і нова мова програмування, для неї вішла велика кількість бібліотек і фреймворків, розглянемо їх:

1) Swift AI - Високооптимізована бібліотека штучного інтелекту та машинного навчання, написана в Swift.
2) Swift for Tensorflow - платформа наступного покоління для машинного навчання, яка включає найновіші дослідження в машинному навчанні, компіляторах, диференційованому програмуванні, дизайні систем та інших.

3) swix - Бібліотека з відкритим кодом, описує деякі функції OpenCV для розробки iOS.
4) MLKit - проста бібліотека машинного навчання, написана в Swift. В даний час функціонують проста лінійна регресія, поліноміальна регресія.
5) Swift Brain - перша бібліотека нейронної мережі / машинного навчання, написана в Swift. Це проект алгоритмів AI у Swift для розробки iOS та OS X. Цей проект включає алгоритми, орієнтовані на теорему Байєса, нейронні мережі, SVM, матриці тощо ...

Хоча C++ і душе швидка мова програмування, але в ній незручно працювати з нейронними мережами через вмалу кількість фреймворків для цього, а також складно обробляти великі об’єми інформації, що являється основою машинного навчання. Тому відкидаємо C++.

Swift хоч і виглядає перспективним для сфери машинного навчання, але ще являється дуже нестабільним, і йому поки що не вистачає інструментів дл обробки даних.

Python стабільний, доволі швидкий, і має всі необхідні інструменти і бібліотеки для задач галузі машинного навчання, тому для розробки системи обробки і класифікації кардіограм було вибрано саме Python.

3.2 Архітектура системи

Зобразимо архітектуру на блок-схемі(рис. 3.2)

Була розроблена згорткова нейронна мережа, яка приймає в якості вхідних даних ЕКГ (вибірки при 200 Гц або 200 зразків в секунду) і видає одне прогнозування кожні 256 кроків(або кожні 1,28 с), які ми називаємо вихідним інтервалом. Мережа приймає за вхід лише зразки ЕКГ та ніяких інших функцій, що стосуються пацієнта чи ЕКГ. Мережева архітектура має 34 шари.(рис 3.3)

Вибіркою для навчання і тестування нейронної мережі було взято MIT-BIH датасет - він налічує приблизно 48 годин сигналів ЕКГ взятих у 47 пацієнтів

між 1975 і 1979 роками. Має як нормальні ЕКГ сигнали, так і сигнали хворих людей.

Було розглянуто 6 класів для серцевого ритму - нормальний(N),передчасне сердцебиття(A),передчасне збудження шлуночків(venticular), тахікардія(paced), комбінація нормального і передчасного збудження шлуночків(F), шуми(не належить до жодного з класів)

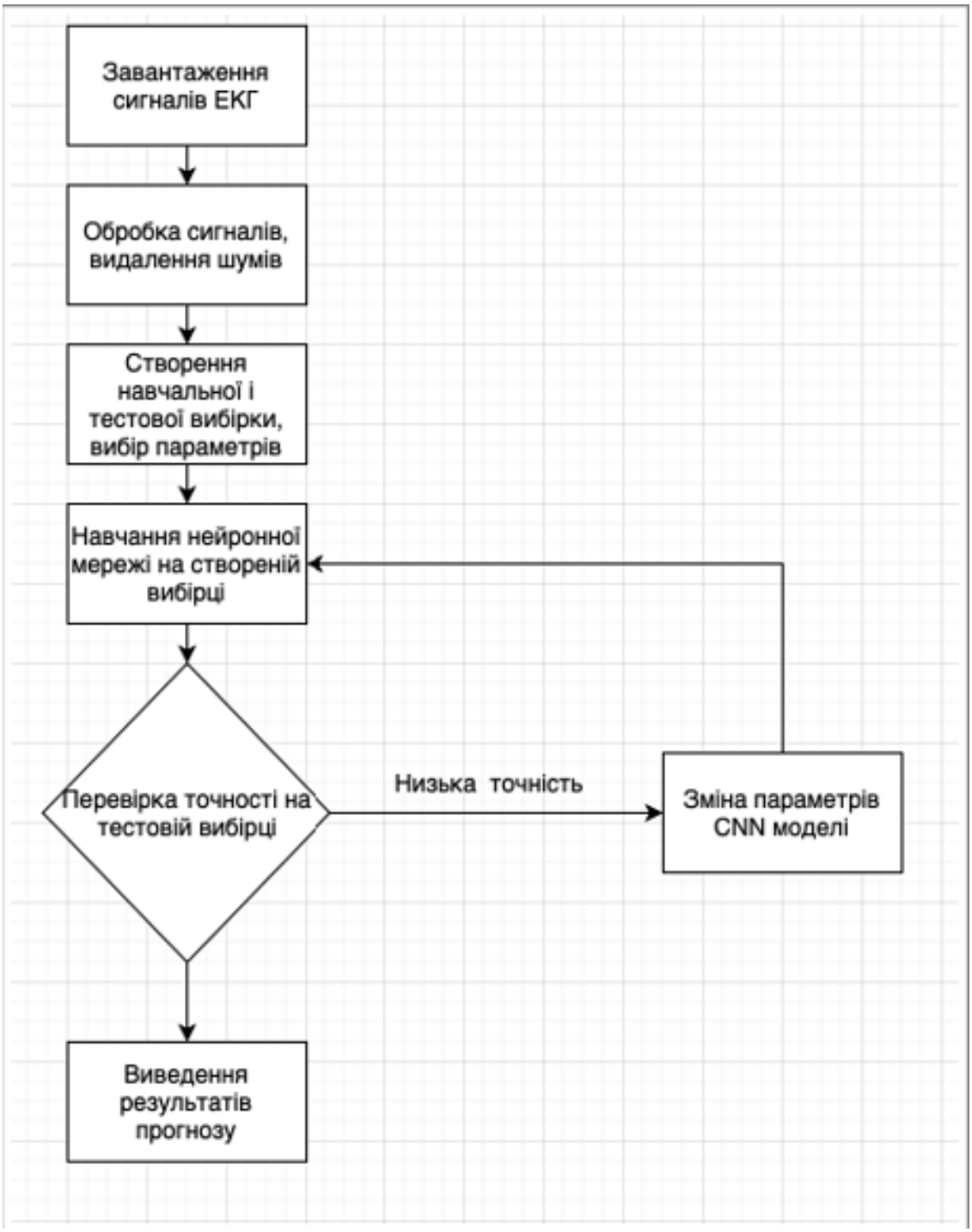


Рисунок 3.2 – Архітектура системи

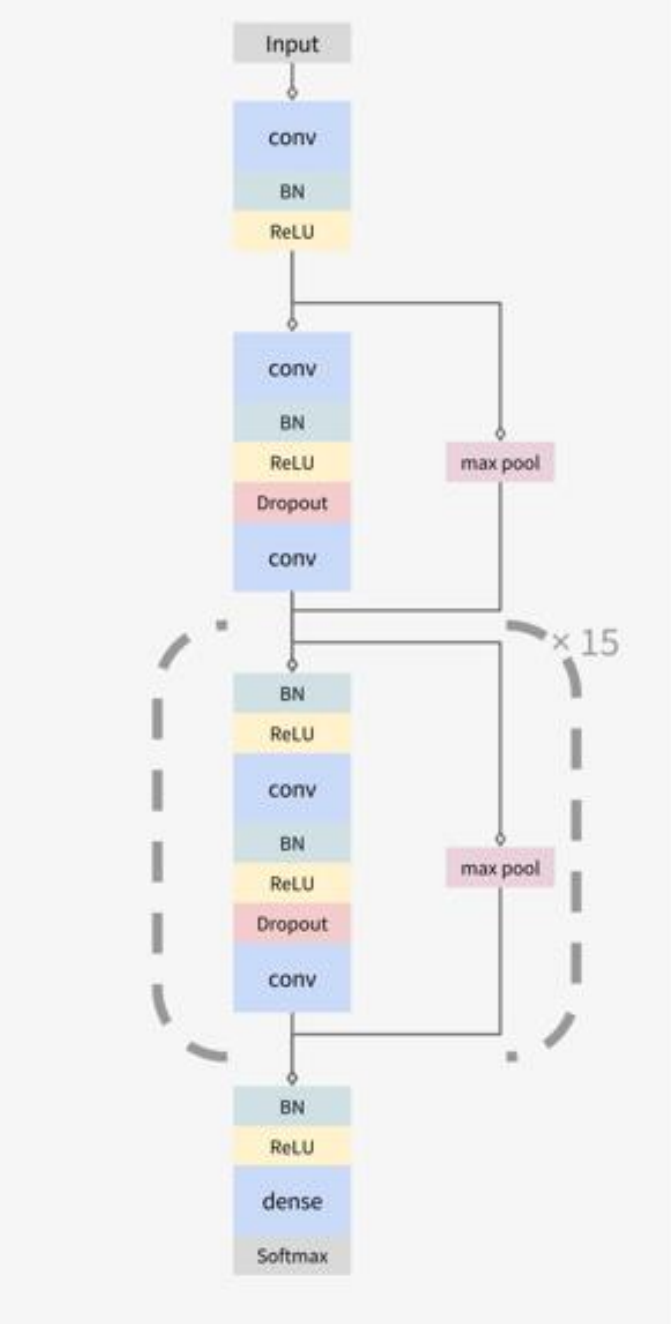


Рисунок 3.3 – Архітектура нейронної мережі

3.3 Аналіз результатів

З матриці помилок, зображеної на рисунку 3.4, видно що модель погано передбачає два класи - A і F, і дуже гарно інші класи. Це сталося через неоднорідність навчальної вибірки, тобто майже 70% сигналів це нормальний ритм, а інші 5 класів лише в 30% вибірки. Через це деякі класи не мають достатньої кількості даних для правильної класифікації.

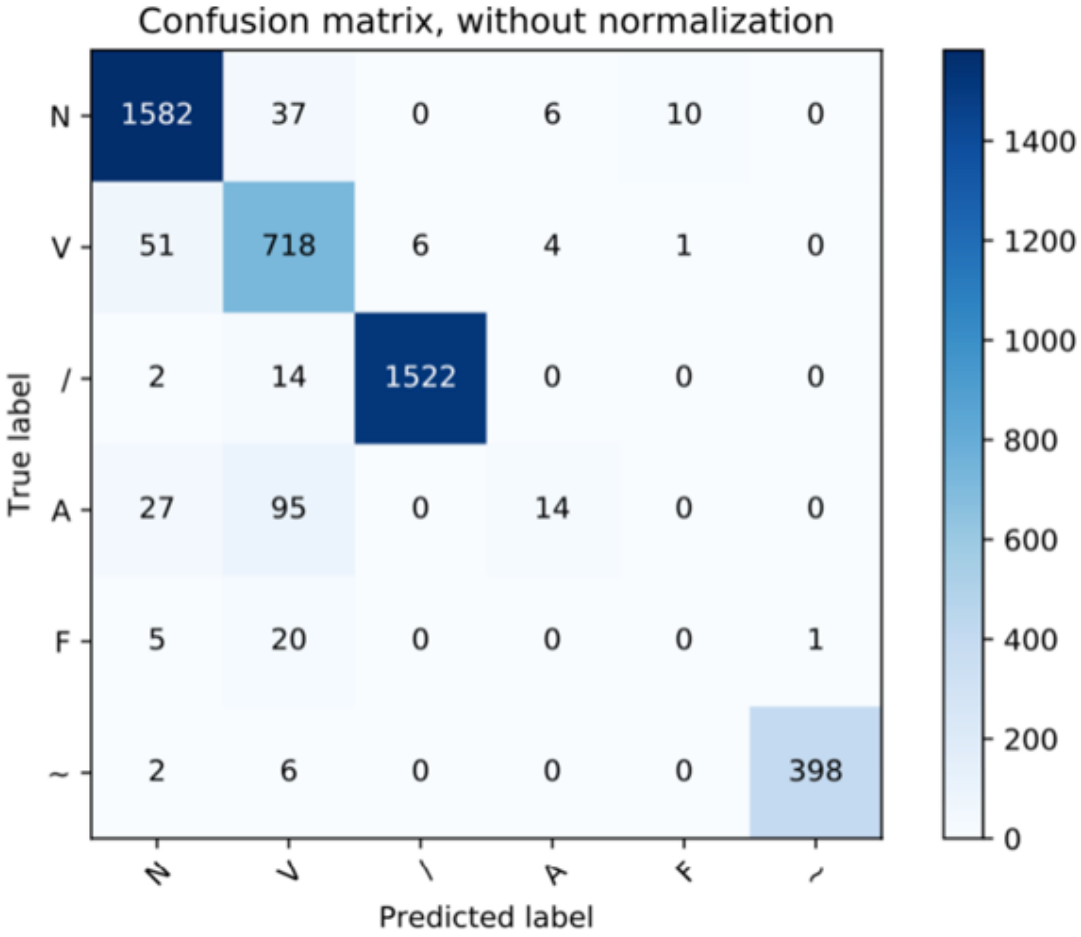


Рисунок 3.4 – Матриця помилок

```
The 17/41-record classified as N with 80.0% certainty
The 18/41-record classified as N with 79.6% certainty
The 19/41-record classified as N with 79.4% certainty
The 20/41-record classified as N with 80.6% certainty
The 21/41-record classified as N with 80.9% certainty
The 22/41-record classified as N with 80.9% certainty
The 23/41-record classified as N with 81.0% certainty
The 24/41-record classified as N with 78.9% certainty
The 25/41-record classified as N with 80.3% certainty
The 26/41-record classified as N with 79.6% certainty
The 27/41-record classified as N with 80.1% certainty
The 28/41-record classified as N with 80.6% certainty
The 29/41-record classified as N with 81.0% certainty
The 30/41-record classified as N with 79.4% certainty
The 31/41-record classified as N with 79.2% certainty
The 32/41-record classified as N with 80.2% certainty
The 33/41-record classified as N with 80.4% certainty
The 34/41-record classified as N with 81.0% certainty
The 35/41-record classified as N with 82.4% certainty
The 36/41-record classified as N with 79.9% certainty
The most predicted label is N with 77.7% certainty
The original label of the record is N
(ecg-env) (base) Air-Kiril:diplomaProj kirill$
```

Рисунок 3.5 – Приклад прогнозу

3.4 Висновки до розділу

У третьому розділі було розглянуто мови програмування, а також їхні бібліотеки і фреймворки з точки зору їх ефективності для проекту який використовує машинне навчання. Для розробки було вибрано мову програмування Python.

Також було проведено аналіз розробленої версії продукту. Була розглянута архітектура системи, будова CNN моделі, точність і матриця помилок фінальної моделі.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Постановка задачі

У даному розділі оцінюємо головні показники результатів кінцевого програмного продукту та його собівартість. Оцінювання проводилося у середовищі PyCharm, мовою програмування було вибрано Python. Нижче приведено декілька різних варіантів модулю і аналіз на вплив цих характеристик на продуктивність роботи програми.

4.2 Обґрунтування функцій дослідження

Основні функції:

F1 - мова програмування

F2 - метод обробки сигналу

F3 - тип нейронної мережі

F4 – вибірка данних для навчання

Функція F1: а) Python; б) C++

Функція F2: а) Фрактальний аналіз; б) Перетворення Фур’є

Функція F3: а) CNN; б) RNN

Функція F4: а) 1000; б) 500

Зобразимо морфологічну карту на рисунку 4.1 і проаналізуємо переваги та недоліки у таблиці 4.1.

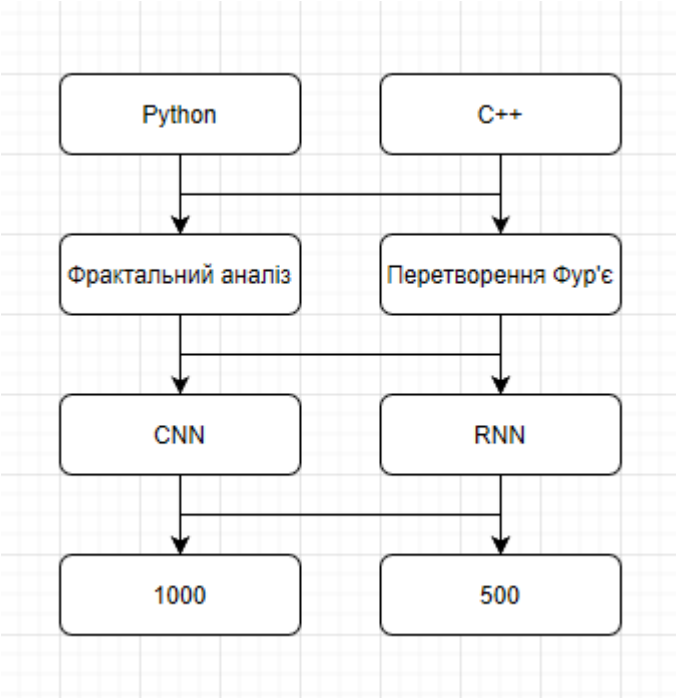


Рисунок 4.1 Морфологічна карта

Таблиця 4.1 - Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	А	Наявність великої кількості бібліотек, які полегшують і пришвидшують використання нейронних мереж	Порівнянно повільна швидкість
	Б	Є багато технічної документації, дуже швидка мова програмування	Майже немає бібліотек пов’язаних з нейронними мережами
F2	А	Доступний для вивчення і реалізації	Довго виконується для великих об’ємів даних
	Б	Швидкий, ефективний	Складний у вивченні

F3	А	Дає високі показники у задачах класифікації	Метод не підтверджено теоретично
	Б	Пристосований для розв’язання даного класу задач	Менш ефективний
F4	А	Більш точні результати	Повільне виконання
	Б	Швидше	Можливі менш точні результати

Роблячи висновок з позитивно-негативної матриці, можемо побачити доцільність використання одних варіантів і недоцільність використання інших.

Зважаючи на результати порівняльного аналізу варіантів реалізації простих функцій, виключаємо варіанти F1b, F2б, F3а. Отже, варіанти які залишилися:

F1a – F2a – F3б – F4a

F1a – F2a – F3б – F4б

Оцінимо якість розглянутих функцій, для цього опишемо систему параметрів.

4.3 Обґрунтування системи параметрів ПП

Для охарактеризування програмного продукту було взято такі параметри:

X1 – швидкість мови програмування(час, витрачений на виконная коду продукту)

X2 – оперативна пам'ять(кількість, необхідна для коректної роботи);

X3 –обробка даних(час, витрачений на обробку даних);

X4 – об’єм написаного коду(розмір коду програми).

Гірші, середні і кращі значення X1,X2,X3,X4 вибираються на вимогах замовника та умов перебігу дослідження, вони наведені у таблиці 4.2:

Таблиця 4.2 - Основні параметри ПП

Умовні позначення	Одиниці виміру	Гірші	Середні	Кращі
X1	Оп/мс	2000	8000	15000
X2	мб	32	16	8
X3	мс	1000	300	120
X4	Кількість рядків коду	3000	2000	1000

Зобразимо X1,X2,X3,X4 на графіках, рисунки 4.2 і 4.3.

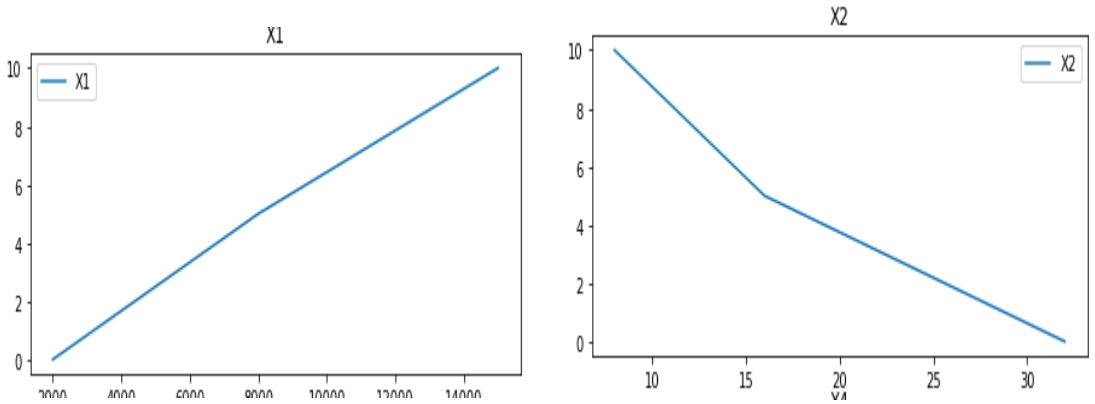


Рисунок 4.2 - зліва X1, справа X2

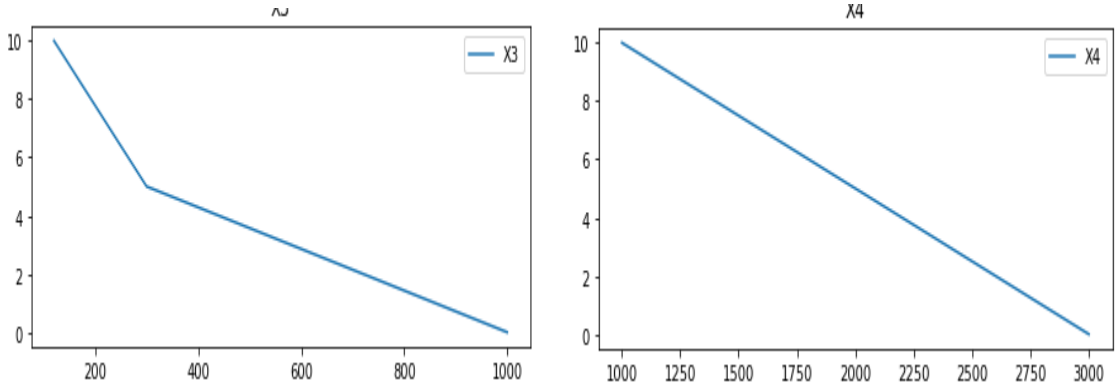


Рисунок 4.3 - зліва X3, справа X4

4.4 Аналіз експертного оцінювання параметрів

Оцінимо значення кожного параметра методом попарного порівняння. Наведемо нижче таблицю 4.3 з результатами

Таблиця 4.3 - Результати ранжування параметрів

Умовне позначення	Одиниці виміру	Ранг експерта 1	2	3	4	5	6	7	Сума рангів R_i	Відхилення Δ_i	Δ_{i2}
X1	Оп/мс	2	1	1	2	1	1	1	9	-8.5	72.25
X2	мб	3	2	3	3	2	3	2	19	1.5	2.25
X3	мс	4	4	4	4	3	4	4	27	9.5	90.25
X4	Кількість рядків коду	1	3	1	1	4	2	3	15	-2.5	6.25
Разом		10	10	10	10	10	10	10	70	0	171

Спочатку порахуємо загальну суму квадратів відхилення

$$S = \sum_{i=1}^{N} \Delta_i^2 = 171 \tag{4.1}$$

де Δ_i^2 – відхилення;

N – число експертів.

Далі коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 171}{7^2(4^3 - 4)} = 0.71 \tag{4.2}$$

Так як обчислений нами коефіцієнт узгодженості більше ніж 0,67(нормативний коефіцієнт), будемо вважати ранжування достовірним.

Будемо вважати 1 ранг найнижчим а 4 – найвищим. Тому експерт поставить найважливішому, на його думку, параметру 4, а найменш важливому відповідно 1. Зобразимо це у таблиці 4.4.

Таблиця 4.4 - Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	>	<	<	<	<	0.5
X1 і X3	<	<	<	<	<	<	<	<	0.5
X1 і X4	>	<	<	<	<	<	<	<	0.5
X2 і X3	<	<	<	<	<	<	<	<	0.5
X2 і X4	>	>	>	<	<	>	<	>	1.5
X3 і X4	>	>	>	>	>	>	>	>	1.5

X3 > X2 > X4 > X1

Виконаємо розрахунок вагомості для кожного параметра і запишемо у таблицю 4.5, а показники рівня якості у таблицю 4.6

Розрахуємо параметр $К_{ві}$:

$$К_{ві} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.3)$$

де $b_i = \sum_{i=1}^{N} a_{ij}$

Починаючи з другого кроку розраховуємо відносні оцінки іншою формулою:

$$К_{ві} = \frac{b_i'}{\sum_{i=1}^{n} b_i'} \quad (4.4)$$

де $b_i' = \sum_{i=1}^{N} a_{ij} b_i$

Розрахуємо вагомості параметрів у таблиці 4.5, і показники рівня якості у таблиці 4.6.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітерація		Друга ітерація		Третя ітерація	
	1	2	3	4	b_i	$K_{\text{ві}}$	b_{i1}	$K_{\text{ві}1}$	b_{i2}	$K_{\text{ві}2}$
X1	1,0	0,5	0,5	0,5	2.5	0.178	6.25	0.122	39.06	0.052
X2	0,5	1,0	0,5	1,5	3.5	0.250	12.25	0.240	150.06	0.200
X3	0,5	0,5	1,0	1,5	3.5	0.250	12.25	0.240	150.06	0.200
X4	0,5	1,5	1,5	1,0	4.5	0.321	20.25	0.397	410.06	0.547
Всього:					14	1	51	1	749.24	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	9000	5.8	0.052	0,3016
F2(X2)	А	16	4.9	0.200	0,98
F3(X3)	Б	360	4.7	0.200	0,94
F4(X4)	А	1900	5.4	0.547	2,9538
	Б	2700	1.3	0.547	0,71

Далі, використовуючи формулу:

$$K_k = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \cdots + K_{\text{ТУ}}[F_{zk}] \qquad (4.5)$$

Обчислимо значення K_{k1} і K_{k2}

$$K_{k1} = 0{,}3016 + 0{,}98 + 0{,}94 + 2{,}9538 = 5{,}1754$$

$$K_{k2} = 0{,}3016 + 0{,}98 + 0{,}94 + 0{,}71 = 2{,}9316$$

Можемо зробити висновок, що перший варіант краще другого, так як для нього коефіцієнт технічного рівня більший.

4.6 Економічний аналіз варіантів розробки ПП

Щоб визначити вартість розробки програмного продукту розрахуємо трудомісткість. Обидва варіанти складаються з 4 завдань :

1.Розробка ПП

2.Розробка програмного інтерфейсу

3.Вибір нейронної мережі

4.Підготовка даних

4.1 Готовий датасет

4.2 Модифікований датасет

У першому і другому варіанті перші три завдання однакові, відрізняється лише 4 завдання, адже від кількості даних залежить і метод їх підготовки. При меншій кількості даних можна знайти готовий датасет, а при більшій, працівникам необхідно буде розмічати дані для нейронної мережі або модифікувати дані для збільшення вибірки.

Для завдання 1 (Алгоритм складності 1, ступінь новизни А, вид використаної інформації - довідкова інформація) T_P =90, $K_П$ = 1.7, $K_{СК}$ = 1,
$K_{СТ}$ = 0.8

T_1 = 90·1.7·0.8 = 122.4 людиноднів

Для завдання 2 (Алгоритм складності 3, ступінь новизни Б, вид використаної інформації – у виді даних) T_P =19, $K_П$ = 1.42, $K_{СК}$ = 1,
$K_{СТ}$ = 0.8

T_2 = 19·1.42·0.8 = 21.584 людиноднів

Для завдання 3 (Алгоритм складності 3, ступінь новизни Б, вид використаної інформації – у виді даних) T_P =19, $K_П$ = 1.42, $K_{СК}$ = 1,

$К_{СТ}$ = 0.8

$Т_3$ = 19·1.42·0.8 = 21.584 людиноднів

Для завдання 4 варіанту 1 (Алгоритм складності 3, ступінь новизни В, вид використаної інформації – БД) $Т_Р$ =12, $К_П$ =0.5, $К_{СК}$ = 1,

$К_{СТ}$ = 0.8

$Т_{4\_1}$ = 12·0.5·0.8 = 4.8 людиноднів

Для завдання 4 варіанту 2 (Алгоритм складності 2, ступінь новизни В, вид використаної інформації – БД) $Т_Р$ =19, $К_П$ =0.6, $К_{СК}$ = 1,

$К_{СТ}$ = 0.6

$Т_{4\_2}$ = 19·0.6·0.6 = 6.91 людиноднів

Обчислимо значення $Т_I$ і $Т_{II}$:

$Т_I$ = (122.4 + 21,584 + 21,584 + 4.8) · 8 = 1362,944 людино-годин.

$Т_{II}$ = (122.4 + 21,584 + 21,584 + 6.91) · 8 = 1379.744 людино-годин.

Бачимо що $Т_{II} > Т_I$

У процесі розробки приймають участь один аналітик з окладом 12000 грн і два розробники з окладом 8000 грн:

$$С_ч = \frac{М}{Т_м * t} грн \quad (4.6)$$

де М- місячний оклад;

$Т_м$- кількість робочих днів тиждень, t – кількість робочих годин в день.

$$С_ч = \frac{8000 + 8000 + 12000}{3 * 21 * 8} = 55.55 грн$$

Далі, знайдемо заробітну плату:

$$С_ч = С_ч Т_і К_д \quad (4.7)$$

І розрахуємо за двома варіантами:

1) $С_{ЗП}$ = 55.55 *1362,944 *1.2 =90853,85 грн.

2) $С_{ЗП}$ = 55.55 *1379,744 *1.2 =91973,74 грн.

Відрахування на соціальний внесок становить 22%:

1) $С_{ВІД}$ = 90853,85*0.22 =19987,847 грн.

2) $С_{ВІД}$ = 91973,74 *0.22 =20234,74 грн.

Далі визначимо $С_м$ – оплату 1 машино-години. Маємо, що 1 ЕОМ обслуговує 1 програміста з окладом 8000 грн., коефіцієнт зайнятості дорівнює 0,2, тоді:

$$C_Г = 12 * 8000 * 0.2 = 19200$$

Тепер врахуємо додаткову заробітню плату:

$$C_{ЗП} = 19200(1 + 0,2) = 23040$$

$$C_{ЗП} = C_{ЗП} * 0,22 = 23040 * 0,22 = 5068,8 \text{ грн.}$$

Амортизаційні відрахування. Амортизація – 20%, та вартість ЕОМ – 10000 грн.

$$C_А = К_{ТМ} * К_А * Ц_{ПР} = 1.15 * 0,2 * 10000 = 2300 \text{ грн} \qquad (4.8)$$

Витрати на профілактику і ремонт:

$$C_Р = К_{ТМ} * К_Р * Ц_{ПР} = 1.15 * 10000 * 0.05 = 575 \text{ грн} \qquad (4.9)$$

Ефективний годинний фонд часу ПК за рік:

$$Т_{ЕФ} = (Д_к - Д_в - Д_С - Д_Р) * t_з * К_в \qquad (4.10)$$

$$Т_{ЕФ} = 365 - 104 - 8 - 16) * 8 * 0.9 = 1706.4$$

Оплата електроенергії:

$Ц_{ЕН} = 1,75$ грн/кВт*год

$$C_{ЕЛ} = Т_{ЕФ} * N_c * К_З * Ц_{ЕН} \qquad (4.11)$$

$$C_{ЕЛ} = 1706.4 * \ 0{,}32 * \ 0.3 * 1.75 \ = \ 286{,}675 \text{ грн}$$

Накладні витрати:

$$C_H = Ц_{ПР} * 0.67 = 10000 * 0.67 = 6700 \text{ грн.} \quad (4.12)$$

Річні експлуатаційні витрати:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н \quad (4.13)$$

$С_{ЕКС} = 23040 \ + \ 5068{,}8 \ + \ 2300 \ + \ 575 \ + \ 212.96 \ + \ 6700 \ =$ 37896,76 грн

Собівартість однієї машино-години ЕОМ:

$$С_М = \frac{С_{ЕКС}}{Т_{ЕФ}} = 38355{,}01 / 1706.4 = 22{,}21 \text{ грн/год.} \quad (4.14)$$

Витрати на оплату машинного часу різних варіантів:

1) $С_М = 22{,}21 \ . \ 1362{,}944 = 30270.99$ грн

2) $С_М = 22{,}21 \ . \ 1379.744 = 30644.11$ грн.

Накладні витрати:

$$C_H = C_{ЗП} * 0.67 \quad (4.15)$$

1) $С_Н = 90853{,}85 . \ 0{,}67 = 60872{,}08$ грн.

2) $С_Н = 91973{,}74 . \ 0{,}67 = 61622{,}41$ грн.

Вартість розробки ПП за варіантами:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н \quad (4.16)$$

1) $С_{ПП}$ = 90853,85 + 19987,84+ 30270.99 + 60872,08 = 201714,76 грн.

2) $С_{ПП}$ = 91973,74 + 20234,74 + 30644.11 + 61622,41 = 204475,18 грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Коефіцієнт техніко-економічного рівня:

$$K_{\text{тер\_i}} = \frac{К_{К\_i}}{С_{Ф\_i}} \quad (4.17)$$

$$K_{\text{тер\_1}} = \frac{5{,}1754}{201714{,}76} = 0{,}2566 * 10^{-4},$$

$$K_{\text{тер\_2}} = \frac{2{,}9316}{204475{,}18} = 0{,}1434 * 10^{-4},$$

Тобто, перший варіант є ефективнішим за другий. $К_{ТЕР1}$= $0{,}2566 * 10^{-4}$.

4.8 Висновки до розділу

У даному розділі, було виконано повний функціонально-вартісний аналіз ПП.

Після виконання першого відбору залишилося лише дві альтернативи, а після виконання повного функціонально-вартісного аналізу залишився найкращий варіант з $К_{ТЕР}$ = $0{,}2566 * 10^{-4}$.

Цей варіант має такі параметри:

- Python
- Фрактальний аналіз
- CNN
- 1000 даних для вибірки

Даний варіант виконання програмного продукту має гарну швидкодію, точно класифікує ЕКГ, не потребує великих об’ємів даних.

ВИСНОВКИ

У ході виконання дипломної роботи було досліджено прикладну частину роботи – будову серця, будову сигналу електрокардіограми, а також зв’язок між ними. Також було розглянуто існуючі системи розпізнавання хворою серця, їх функціонал.

У другому розділі було розглянуто нейронні мережі, а саме згорткову нейронну мережу і рекурентну нейронну мережу, визначено їх переваги та недоліки. У результаті було вибрано CNN як модель яка використовується у кінцевому продукті. Також було розглянуто два методи обробки сигналів – перетворення Фур’є і фрактальний метод. Були описані алгоритми роботи цих методів.

Виконано аналіз декількох мов програмування і їхніх бібліотек і фреймворків для визначення найбільш підходящої для розробки продукту, їх сильні і слабкі сторони. У результаті дослідження перевагу було вибрано за основу мову програмування Python, через наявність великої кількості бібліотек і фреймворків для машинного навчання.

Було створену архітектуру згорткової нейронної мережі, і знайдено датасет для використання у навчанні. В результаті була отримана модель, яка аналізує вхідний сигнал ЕКГ і визначає наявність хвороби серця у людини. Також було проведено тестування і аналіз робочої моделі – визначено її точність а також матрицю помилок.

ПЕРЕЛІК ПОСИЛАНЬ

1. Long Short-Term Memory [Електронний ресурс] – URL: http://www.bioinf.jku.at/publications/older/2604.pdf (дата звернення 22.05.2020)
2. A Survey of the Recent Architectures of Deep Convolutional Neural Networks [Електронний ресурс] – URL: https://arxiv.org/pdf/1901.06032.pdf (дата звернення 22.05.2020)
3. Штучні нейронні мережі :обчислення [Електронний ресурс] – URL: http://www.immsp.kiev.ua/postgraduate/Biblioteka_trudy/ShtuchnNejronMeregNester2004.pdf (дата звернення 22.05.2020)
4. A Comprehensive Guide to Convolutional Neural Networks [Електронний ресурс] – URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (дата звернення 22.05.2020)
5. Пашин В.П. Функционально-стоимостный анализ конструкторскотехнологических решений. - К.: РДЭНТП «Знание» УССР, 1989. - 22с.
6. Пашін В.П. Оцінка конкурентоспроможності електронних пристроїв на стадії проектування. - К. Економічний вісник НТУУ „КПІ", 2006. - №3. с. 252-255.
7. Пашин В.П. Управление качеством изделий на основе функциональностоимостного анализа. - К.: «Технология и организация производства», 1989. - №1. с. 17-19.
8. Classification of 7 Arrhythmias from ECG Using Fractal Dimensions [Електронний ресурс] – URL: http://www.fortunejournals.com/articles/classification-of-7-arrhythmias-from-ecg-using-fractal-dimensions.html (дата звернення 22.05.2020)
9. Fractal Character of the Electrocardiogram: Distinguishing Heart-Failure and Normal Patients [Електронний ресурс] – URL: https://pubmed.ncbi.nlm.nih.gov/8678358/ (дата звернення 22.05.2020)
10. Signal Processing Techniques for Removing Noise from ECG Signals [Електронний ресурс] – URL: http://www.jscholaronline.org/articles/JBER/Signal-Processing.pdf (дата звернення 22.05.2020)
11. ELECTROCARDIOGRAM (ECG) SIGNAL PROCESSING [Електронний ресурс] – URL: http://diec.unizar.es/intranet/articulos/uploads/libroWiley.pdf.pdf (дата звернення 22.05.2020)

12. A Comprehensive Guide to Convolutional Neural Networks [Електронний ресурс] – URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (дата звернення 22.05.2020)
13. Convolutional Neural Network Architecture: Forging Pathways to the Future [Електронний ресурс] – URL: https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-architecture-forging-pathways-future/ (дата звернення 22.05.2020)
14. Fundamentals of Deep Learning – Introduction to Recurrent Neural Networks [Електронний ресурс] – URL: https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/ (дата звернення 22.05.2020)
15. Recurrent Neural Network [Електронний ресурс] – URL: https://www.sciencedirect.com/topics/engineering/recurrent-neural-network (дата звернення 22.05.2020)
16. Convolutional Neural Network (CNN): Graphical Visualization [Електронний ресурс] – URL: https://www.analyticssteps.com/blogs/convolutional-neural-network-cnn-graphical-visualization-code-explanation (дата звернення 22.05.2020)
17. Analyzing Electrocardiography (ECG) Signal using Fractal Method [Електронний ресурс] – URL: http://inpressco.com/wp-content/uploads/2017/04/Paper27498-505.pdf (дата звернення 22.05.2020)
18. Non-linear dynamics and fractal composition of human electrocardiogram: a proposed universal formula for ECG waveforms [Електронний ресурс] – URL: https://www.oatext.com/Non-linear-dynamics-and-fractal-composition-of-human-electrocardiogram-a-proposed-universal-formula-for-ECG-waveforms.php#gsc.tab=0 (дата звернення 22.05.2020)

Додаток А Лістинг Програми

```
def train(config, X, y, Xval=None, yval=None):


    classes = ['N','V','/','A','F','~']
    Xe = np.expand_dims(X, axis=2)
    if not config.split:
        from sklearn.model_selection import train_test_split
        Xe, Xvale, y, yval = train_test_split(Xe, y, test_size=0.2, random_state=1)
    else:
        Xvale = np.expand_dims(Xval, axis=2)
        (m, n) = y.shape
        y = y.reshape((m, 1, n ))
        (mvl, nvl) = yval.shape
        yval = yval.reshape((mvl, 1, nvl))


    if config.checkpoint_path is not None:
        model = model.load_model(config.checkpoint_path)
        initial_epoch = config.resume_epoch
    else:
        model = ECG_model(config)
        initial_epoch = 0


    mkdir_recursive('models')
    callbacks = [
            EarlyStopping(patience = config.patience, verbose=1),
            ReduceLROnPlateau(factor = 0.5, patience = 3, min_lr = 0.01, verbose=1),
            TensorBoard( log_dir='./logs', histogram_freq=0, write_graph = True, write_grads=False,
write_images=True),
            ModelCheckpoint('models/{}-latest.hdf5'.format(config.feature), monitor='val_loss',
save_best_only=False, verbose=1, period=10)
            # , lr_decay_callback
    ]


    model.fit(Xe, y,
            validation_data=(Xvale, yval),
```

```
            epochs=config.epochs,
            batch_size=config.batch,
            callbacks=callbacks,
            initial_epoch=initial_epoch)
    print_results(config, model, Xvale, yval, classes, )


def main(config):
    print('feature:', config.feature)
    (X,y, Xval, yval) = loaddata(config.input_size, config.feature)
    rain(config, X, y, Xval, yval)

if __name__=="__main__":
    config = get_config()
    main(config)
def cincData(config):
    num = config.num
    import csv
    testlabel = []

    with open('training2017/REFERENCE.csv') as csv_file:
      csv_reader = csv.reader(csv_file, delimiter=',')
      line_count = 0
      for row in csv_reader:
        testlabel.append([row[0],row[1]])
        #print(row[0], row[1])
        line_count += 1
      print(f'Processed {line_count} lines.')
    if num == None:
      high = len(testlabel)-1
      num = np.random.randint(1,high)
    filename , label = testlabel[num-1]
    filename = 'training2017/'+ filename + '.mat'
    from scipy.io import loadmat
```

```
    data = loadmat(filename)
    print("The record of "+ filename)
    if not config.upload:
        data = data['val']
        _, size = data.shape
        data = data.reshape(size,)
    else:
        data = np.array(data)
    return data, label


def predict(data, label, peaks, config):
    classesM = ['N','Ventricular','Paced','A','F','Noise']
    predicted, result  = predictByPart(data, peaks)
    sumPredict = sum(predicted[x][1] for x in range(len(predicted)))
    avgPredict = sumPredict/len(predicted)
    print("The average of the predict is:", avgPredict)
    print("The most predicted label is {} with {:3.1f}% certainty".format(classesM[avgPredict.argmax()],
100*max(avgPredict[0])))
    sec_idx = avgPredict.argsort()[0][-2]
    print("The second predicted label is {} with {:3.1f}% certainty".format(classesM[sec_idx],
100*avgPredict[0][sec_idx]))
    print("The original label of the record is " + label)
    if config.upload:
      return predicted, classesM[avgPredict.argmax()], 100*max(avgPredict[0])


def predictByPart(data, peaks):
    classesM = ['N','Ventricular','Paced','A','F','Noise']#,'L','R','f','j','E','a','J','Q','e','S']
    predicted = list()
    result = ""
    counter = [0]* len(classesM)
    from keras.models import load_model
    model = load_model('models/MLII-latest.hdf5')
    config = get_config()
    for i, peak in enumerate(peaks[3:-1]):
      total_n =len(peaks)
```

```
      start, end =  peak-config.input_size//2 , peak+config.input_size//2
      prob = model.predict(data[:, start:end])
      prob = prob[:,0]
      ann = np.argmax(prob)
      counter[ann]+=1
      if classesM[ann] != "N":
        print("The {}/{}-record classified as {} with {:3.1f}%
certainty".format(i,total_n,classesM[ann],100*prob[0,ann]))
      result += "("+ classesM[ann] +":" + str(round(100*prob[0,ann],1)) + "%)"
      predicted.append([classesM[ann],prob])
      if classesM[ann] != 'N' and prob[0,ann] > 0.95:
        import matplotlib.pyplot as plt
        plt.plot(data[:, start:end][0,:,0],)
        mkdir_recursive('results')
        plt.savefig('results/hazard-'+classesM[ann]+'.png', format="png", dpi = 300)
        plt.close()
    result += "{}-N, {}-Venticular, {}-Paced, {}-A, {}-F, {}-Noise".format(counter[0], counter[1],
counter[2], counter[3], counter[4], counter[5])
    return predicted, result


def main(config):
  classesM= ['N','Ventricular','Paced','A','F', 'Noise']#,'L','R','f','j','E','a','J','Q','e','S']


  if config.upload:
    data = uploadedData(file)
  else:
    data, label = cincData(config)
  data, peaks = preprocess(data, config)
  return predict(data, label, peaks, config)


if __name__=='__main__':
  config = get_config()
  main(config)
def preprocess( split ):
```

```
    nums =
['100','101','102','103','104','105','106','107','108','109','111','112','113','114','115','116','117','118','119','121'
,'122','123','124','200','201','202','203','205','207','208','209','210','212','213','214','215','217','219','220','221',
'222','223','228','230','231','232','233','234']
    features = ['MLII', 'V1', 'V2', 'V4', 'V5']


    if split :
        testset = ['101', '105','114','118', '124', '201', '210' , '217']
        trainset = [x for x in nums if x not in testset]


    def dataSaver(dataSet, datasetname, labelsname):
        classes = ['N','V','/','A','F','~']#,'L','R',f','j','E','a']#,'J','Q','e','S']
        Nclass = len(classes)
        datadict, datalabel= dict(), dict()


        for feature in features:
            datadict[feature] = list()
            datalabel[feature] = list()


        def dataprocess():
          input_size = config.input_size
          for num in tqdm(dataSet):
            from wfdb import rdrecord, rdann
            record = rdrecord('dataset/'+ num, smooth_frames= True)
            from sklearn import preprocessing
            signals0 = preprocessing.scale(np.nan_to_num(record.p_signal[:,0])).tolist()
            signals1 = preprocessing.scale(np.nan_to_num(record.p_signal[:,1])).tolist()
            from scipy.signal import find_peaks
            peaks, _ = find_peaks(signals0, distance=150)


            feature0, feature1 = record.sig_name[0], record.sig_name[1]


            global lppened0, lappend1, dappend0, dappend1
            lappend0 = datalabel[feature0].append
            lappend1 = datalabel[feature1].append
```

```
            dappend0 = datadict[feature0].append
            dappend1 = datadict[feature1].append
            # skip a first peak to have enough range of the sample
            for peak in tqdm(peaks[1:-1]):
              start, end =  peak-input_size//2 , peak+input_size//2
              ann = rdann('dataset/'+ num, extension='atr', sampfrom = start, sampto = end,
return_label_elements=['symbol'])

              def to_dict(chosenSym):
                y = [0]*Nclass
                y[classes.index(chosenSym)] = 1
                lappend0(y)
                lappend1(y)
                dappend0(signals0[start:end])
                dappend1(signals1[start:end])

              annSymbol = ann.symbol
              # remove some of "N" which breaks the balance of dataset
              if len(annSymbol) == 1 and (annSymbol[0] in classes) and (annSymbol[0] != "N" or
np.random.random()<0.15):
                to_dict(annSymbol[0])

        dataprocess()
        noises = add_noise(config)
        for feature in ["MLII", "V1"]:
            d = np.array(datadict[feature])
            if len(d) > 15*10**3:
                n = np.array(noises["trainset"])
            else:
                n = np.array(noises["testset"])
            datadict[feature]=np.concatenate((d,n))
            size, _  = n.shape
            l = np.array(datalabel[feature])
            noise_label = [0]*Nclass
            noise_label[-1] = 1
```

```
            noise_label = np.array([noise_label] * size)
            datalabel[feature] = np.concatenate((l, noise_label))
        import deepdish as dd
        dd.io.save(datasetname, datadict)
        dd.io.save(labelsname, datalabel)

    if split:
        dataSaver(trainset, 'dataset/train.hdf5', 'dataset/trainlabel.hdf5')
        dataSaver(testset, 'dataset/test.hdf5', 'dataset/testlabel.hdf5')
    else:
        dataSaver(nums, 'dataset/targetdata.hdf5', 'dataset/labeldata.hdf5')


def main(config):
    def Downloadmitdb():
        ext = ['dat', 'hea', 'atr']
        nums =
['100','101','102','103','104','105','106','107','108','109','111','112','113','114','115','116','117','118','119','121'
,'122','123','124','200','201','202','203','205','207','208','209','210','212','213','214','215','217','219','220','221',
'222','223','228','230','231','232','233','234']
        for num in tqdm(nums):
            for e in ext:
                url = "https://physionet.org/physiobank/database/mitdb/"
                url = url + num +"."+e
                mkdir_recursive('dataset')
                cmd = "cd dataset && curl -O "+url
                os.system(cmd)

    if config.downloading:
        Downloadmitdb()
    return preprocess(config.split)

if __name__=="__main__":
    config = get_config()
    main(config)
```

```
def mkdir_recursive(path):
  if path == "":
    return
  sub_path = os.path.dirname(path)
  if not os.path.exists(sub_path):
    mkdir_recursive(sub_path)
  if not os.path.exists(path):
    print("Creating directory " + path)
    os.mkdir(path)

def loaddata(input_size, feature):
    import deepdish.io as ddio
    mkdir_recursive('dataset')
    trainData = ddio.load('dataset/train.hdf5')
    testlabelData= ddio.load('dataset/trainlabel.hdf5')
    X = np.float32(trainData[feature])
    y = np.float32(testlabelData[feature])
    att = np.concatenate((X,y), axis=1)
    np.random.shuffle(att)
    X , y = att[:,:input_size], att[:, input_size:]
    valData = ddio.load('dataset/test.hdf5')
    vallabelData= ddio.load('dataset/testlabel.hdf5')
    Xval = np.float32(valData[feature])
    yval = np.float32(vallabelData[feature])
    return (X, y, Xval, yval)

class LearningRateSchedulerPerBatch(LearningRateScheduler):
    """ code from https://towardsdatascience.com/resuming-a-training-process-with-keras-3e93152ee11a
    Callback class to modify the default learning rate scheduler to operate each batch"""
    def __init__(self, schedule, verbose=0):
        super(LearningRateSchedulerPerBatch, self).__init__(schedule, verbose)
        self.count = 0  # Global batch index (the regular batch argument refers to the batch index within the epoch)

    def on_epoch_begin(self, epoch, logs=None):
```

```
        pass

    def on_epoch_end(self, epoch, logs=None):
        pass

    def on_batch_begin(self, batch, logs=None):
        super(LearningRateSchedulerPerBatch, self).on_epoch_begin(self.count, logs)

    def on_batch_end(self, batch, logs=None):
        super(LearningRateSchedulerPerBatch, self).on_epoch_end(self.count, logs)
        self.count += 1



def plot_confusion_matrix(y_true, y_pred, classes, feature,
                          normalize=False,
                          title=None,
                          cmap=plt.cm.Blues):
    """Modification from code at https://scikit-
learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html"""
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'

    cm = confusion_matrix(y_true, y_pred)
    #classes = classes[unique_labels(y_true, y_pred)]

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)
```

```
    fig, ax = plt.subplots()
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    ax.figure.colorbar(im, ax=ax)
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           xticklabels=classes, yticklabels=classes,
           title=title,
           ylabel='True label',
           xlabel='Predicted label')

    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
             rotation_mode="anchor")

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, format(cm[i, j], fmt),
                    ha="center", va="center",
                    color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
    mkdir_recursive('results')
    fig.savefig('results/confusionMatrix-'+feature+'.eps', format='eps', dpi=1000)
    return ax


# Precision-Recall curves and ROC curves for each class
def PR_ROC_curves(ytrue, ypred, classes, ypred_mat):
    ybool = ypred == ytrue
    f, ax = plt.subplots(3,4,figsize=(10, 10))
    ax = [a for i in ax for a in i]

    e = -1
    for c in classes:
```

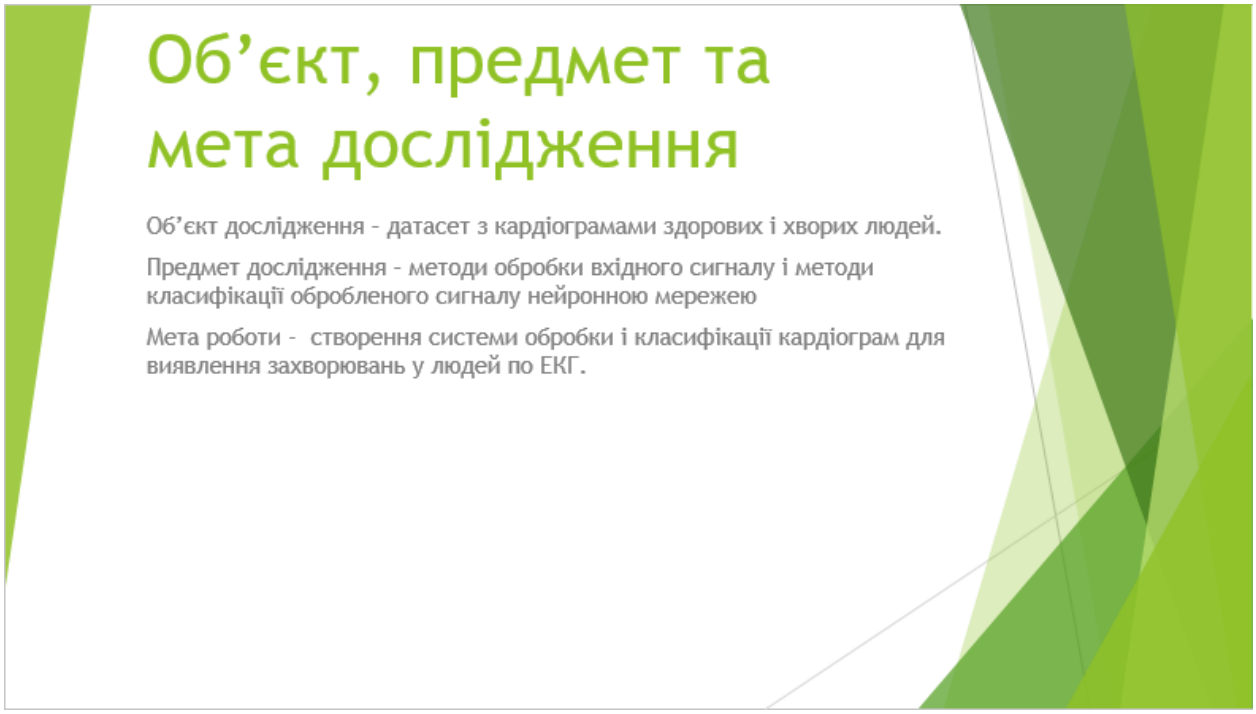
```
idx1 = [n for n,x in enumerate(ytrue) if classes[x]==c]
idx2 = [n for n,x in enumerate(ypred) if classes[x]==c]
idx = idx1+idx2
if idx == []:
    continue
bi_ytrue = ytrue[idx]
bi_prob = ypred_mat[idx, :]
bi_ybool = np.array(ybool[idx])
bi_yscore = np.array([bi_prob[x][bi_ytrue[x]] for x in range(len(idx))])
try:
    print("AUC for {}: {}".format(c, roc_auc_score(bi_ybool+0, bi_yscore)))
    e+=1
except ValueError:
    continue
ppvs, senss, thresholds = precision_recall_curve(bi_ybool, bi_yscore)
cax = ax[2*e]
cax.plot(ppvs, senss, lw=2, label="Model")
cax.set_xlim(-0.008, 1.05)
cax.set_ylim(0.0, 1.05)
cax.set_title("Class {}".format(c))
cax.set_xlabel('Sensitivity (Recall)')
cax.set_ylabel('PPV (Precision)')
cax.legend(loc=3)

fpr, tpr, thresholds = roc_curve(bi_ybool, bi_yscore)
cax2 = ax[2*e+1]
cax2.plot(fpr, tpr, lw=2, label="Model")
cax2.set_xlim(-0.1, 1.)
cax2.set_ylim(0.0, 1.05)
cax2.set_title("Class {}".format(c))
cax2.set_xlabel('1 - Specificity')
cax2.set_ylabel('Sensitivity')
cax2.legend(loc=4)
```

```
    mkdir_recursive("results")
    plt.savefig("results/model_prec_recall_and_roc.eps",
        dpi=400,
        format='eps',
        bbox_inches='tight')
    plt.close()

def print_results(config, model, Xval, yval, classes):
    model2 = model
    if config.trained_model:
        model.load_weights(config.trained_model)
    else:
        model.load_weights('models/{}-latest.hdf5'.format(config.feature))
    # to combine different trained models. On testing
    if config.ensemble:
        model2.load_weight('models/weights-V1.hdf5')
        ypred_mat = (model.predict(Xval) + model2.predict(Xval))/2
    else:
        ypred_mat = model.predict(Xval)
    ypred_mat = ypred_mat[:,0]
    yval = yval[:,0]

    ytrue = np.argmax(yval,axis=1)
    yscore = np.array([ypred_mat[x][ytrue[x]] for x in range(len(yval))])
    ypred = np.argmax(ypred_mat, axis=1)
    print(classification_report(ytrue, ypred))
    plot_confusion_matrix(ytrue, ypred, classes, feature=config.feature, normalize=False)
    print("F1 score:", f1_score(ytrue, ypred, average=None))
    PR_ROC_curves(ytrue, ypred, classes, ypred_mat)

def add_noise(config):
    noises = dict()
    noises["trainset"] = list()
    noises["testset"] = list()
```

```
    import csv
    try:
        testlabel = list(csv.reader(open('training2017/REFERENCE.csv')))
    except:
        cmd = "curl -O https://archive.physionet.org/challenge/2017/training2017.zip"
        os.system(cmd)
        os.system("unzip training2017.zip")
        testlabel = list(csv.reader(open('training2017/REFERENCE.csv')))
    for i, label in enumerate(testlabel):
      if label[1] == '~':
        filename = 'training2017/'+ label[0] + '.mat'
        from scipy.io import loadmat
        noise = loadmat(filename)
        noise = noise['val']
        _, size = noise.shape
        noise = noise.reshape(size,)
        noise = np.nan_to_num(noise) # removing NaNs and Infs
        from scipy.signal import resample
        noise= resample(noise, int(len(noise) * 360 / 300) ) # resample to match the data sampling rate
360(mit), 300(cinc)
        from sklearn import preprocessing
        noise = preprocessing.scale(noise)
        noise = noise/1000*6 # rough normalize, to be improved
        from scipy.signal import find_peaks
        peaks, _ = find_peaks(noise, distance=150)
        choices = 10 # 256*10 from 9000
        picked_peaks = np.random.choice(peaks, choices, replace=False)
        for j, peak in enumerate(picked_peaks):
          if peak > config.input_size//2 and peak < len(noise) - config.input_size//2:
            start,end  = peak-config.input_size//2, peak+config.input_size//2
            if i > len(testlabel)/6:
              noises["trainset"].append(noise[start:end].tolist())
            else:
              noises["testset"].append(noise[start:end].tolist())
    return noises
```

```
def preprocess(data, config):
    sr = config.sample_rate
    if sr == None:
      sr = 300
    data = np.nan_to_num(data) # removing NaNs and Infs
    from scipy.signal import resample
    data = resample(data, int(len(data) * 360 / sr) ) # resample to match the data sampling rate 360(mit), 300(cinc)
    from sklearn import preprocessing
    data = preprocessing.scale(data)
    from scipy.signal import find_peaks
    peaks, _ = find_peaks(data, distance=150)
    data = data.reshape(1,len(data))
    data = np.expand_dims(data, axis=2) # required by Keras
    return data, peaks


# predict
def uploadedData(filename, csvbool = True):
    if csvbool:
      csvlist = list()
      with open(filename, 'r') as csvfile:
        for e in csvfile:
          if len(e.split()) == 1 :
            csvlist.append(float(e))
          else:
            csvlist.append(e)
    return csvlist
```

Додаток Б Ілюстративний матеріал



Постановка задачі

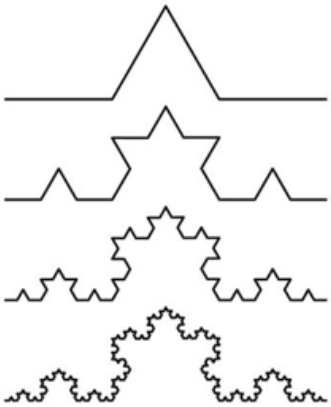
- Розглянути предметну область(електрокардіограма, будова і робота серця)
- Розглянути сучасні методи обробки ЕКГ
- Розглянути методи навчання нейронних мереж і вибрати найкращий для поставленої мети
- Знайти і обробити дані для навчання нейронної мережі
- Провести навчання і тренування моделі
- Протестувати фінальну версію програми

Предметна область, ЕКГ

- Електрокардіограма - крива, яка відображає електричну активність серця.

Обробка часового ряду. Фрактальний метод

- Фрактал - це самоподібна підмножина евклідового простору
- Одною з головних властивостей фрактала вважається самоподібність - інваріантність щодо переносів і скейлінга (зміни масштабу). Це означає, що фрактал в будь-якій мірі одноманітно влаштований у великому діапазоні масштабів



Обробка часового ряду. Фрактальний метод

- Ми можемо розглядати фрактальні властивості вхідного сигналу ЕКГ, і на основі цього визначати захворювання у людей, основним параметром при цьому являється фрактальна розмірність кривої. Тобто у здорової людини і у людини яка має різні захворювання серця фрактальна розмірність сигналу ЕКГ лежить у різних діапазонах.

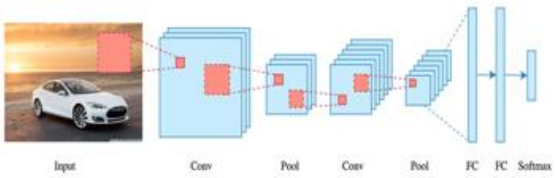
Сигнал	Фрактальна розмірність
Серцевий блок	0.0859 - 0.1229
Фібриляція передсердь	0.1401- 0.3795
Надшлуночкова аритмія	0.3806 -0.5175
Нормальний серцевий ритм	1.1617 - 1.7882

Обробка часового ряду. Перетворення Фур'є

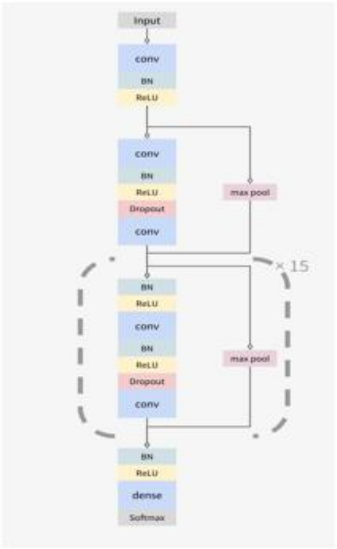
- Будь який періодичний дискретний сигнал можна розкласти в кінцевий ряд Фур'є.
- Вхідний сигнал ми можемо представити у вигляді синусоїд і косинусоїд з різними амплітудами.

Згорткова нейронна мережа(CNN)

1) Згортка
2) Пулінг
3) ANN



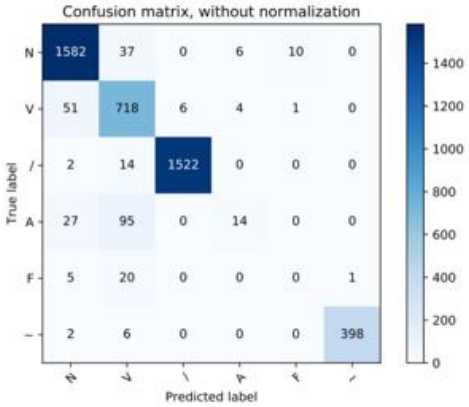
Архітектура CNN моделі



Приклад роботи програми

```
The 17/41-record classified as N with 80.0% certainty
The 18/41-record classified as N with 79.6% certainty
The 19/41-record classified as N with 79.4% certainty
The 20/41-record classified as N with 80.6% certainty
The 21/41-record classified as N with 80.9% certainty
The 22/41-record classified as N with 80.9% certainty
The 23/41-record classified as N with 81.0% certainty
The 24/41-record classified as N with 78.9% certainty
The 25/41-record classified as N with 80.3% certainty
The 26/41-record classified as N with 79.6% certainty
The 27/41-record classified as N with 80.1% certainty
The 28/41-record classified as N with 80.6% certainty
The 29/41-record classified as N with 81.0% certainty
The 30/41-record classified as N with 79.4% certainty
The 31/41-record classified as N with 79.2% certainty
The 32/41-record classified as N with 80.2% certainty
The 33/41-record classified as N with 80.4% certainty
The 34/41-record classified as N with 81.0% certainty
The 35/41-record classified as N with 82.4% certainty
The 36/41-record classified as N with 79.9% certainty
The most predicted label is N with 77.7% certainty
The original label of the record is N
(ecg-env) (base) Air-Kiril:diplomaProj kirill$
```

Аналіз результатів роботи



Дякую за увагу!